ORIC-1

Manual Básico de Programación



ORIC-1

Manual Básico de Programación

por **John Scrive**

Editado y producido para Productos Internacionales Oric S.L. por Publicaciones Sunshine S.L., Londres.



CAPITULO 1
Introducción Página 5

CAPITULO 2
Puesta en marcha de la computadora Página 9
Una guía para poner en marcha su computadora

CAPITULO 3
Programar en Basic Página 15
Aprendiendo el lenguaje de Oric

CAPITULO 4
Color y Gráficos Página 31
El Oric trabaja en cuatro modos y ocho colores

CAPITULO 5
Editar Programas en Basic Página 47
Hay unos comandos de Edición muy potentes
para ayudarle a escribir sus propios programas

CAPITULO 6
Empaquetado de números Página 53
Su Oric también posee una herramienta
matemática muy potente

CAPITULO 7
Más funciones matemáticas Página 69
Una guía para trigonometría y álgebra

CAPITULO 8
Palabras Página 77
Manejo de palabras en cadenas

CAPITULO 9
Gráficos muy Avanzados Página 85
Dibujos de alta resolución y definición de
sus propios caracteres

CAPITULO 10
Sonido Página 95
El Oric tiene cuatro canales separados de sonido y cuatro sonidos predefinidos para juegos

CAPITULO 11
Salvar los programas en el casette Página 103
Una guía para el sistema operativo del casette

CAPITULO 12
Mejor Basic Página 109
Con un poco de práctica y cuidado Vd. puede mejorar sus programas en Basic

CAPITULO 13
Programas en código de máquina Página 121
Introducción del código de máquina

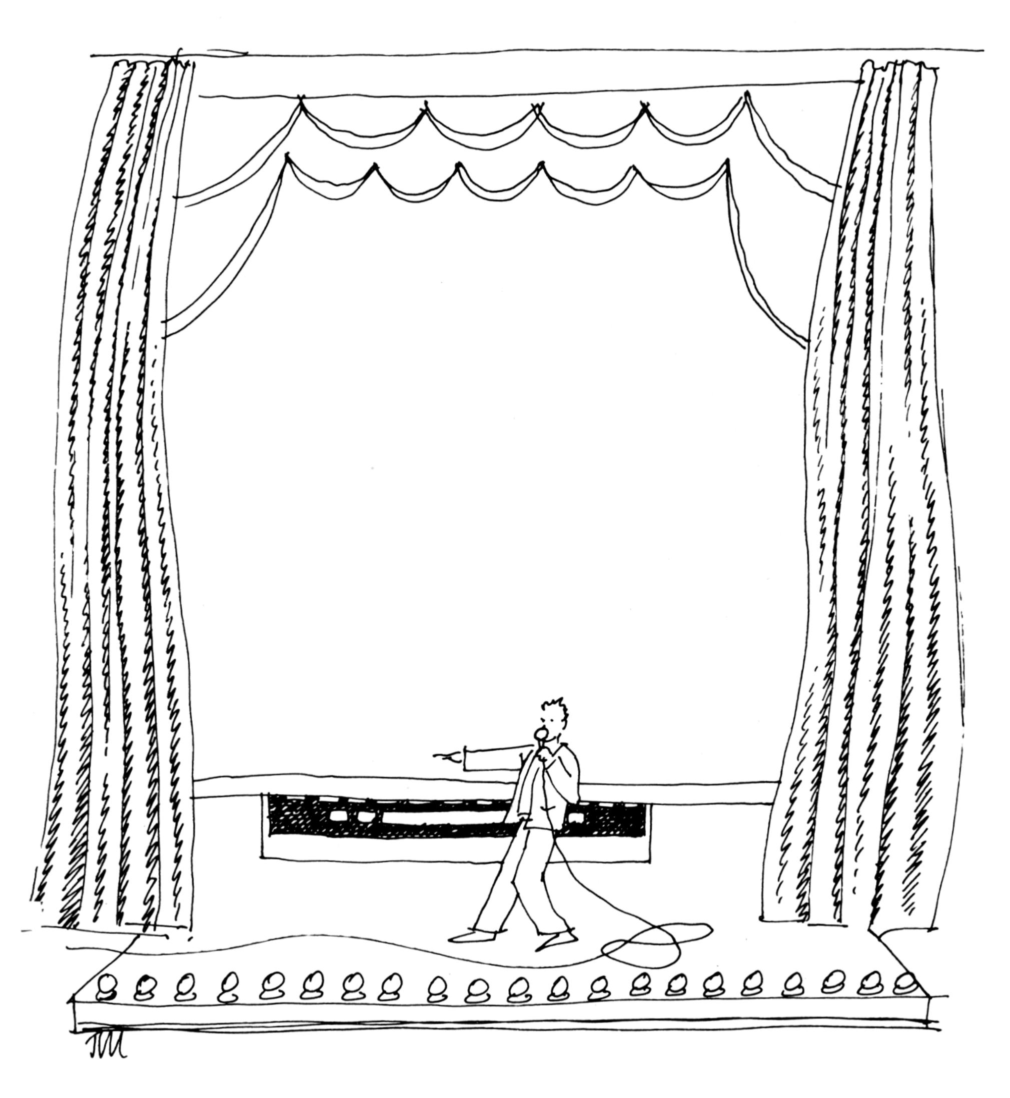
CAPITULO 14
Uso de una impresora Página 131

CAPITULO 15
Basic del Oric Página 135
Una lista de todos los comandos del Basic del Oric

APENDICES

A	**Mapa de la memoria**	Página	145
B	**Caracteres de Control**	Página	146
C	**Atributos**	Página	147
D	**Tabla del ASCII**	Página	148
E	**Tabla Binaria/Hex/Decimal**	Página	149
F	**Tabla pines de los conectores**	Página	151
G	**Funciones Derivadas**	Página	152
H	**Mapa de la pantalla de texto**	Página	154
I	**Mapa de la pantalla de alta resolución**	Página	155
J	**Códigos de error**	Página	156
K	**El Monitor del 6502**	Página	158

CAPITULO 1
Introducción



¡Felicidades!. Vd. es el poseedor de una de las micro-computadoras más avanzadas de hoy en día. Los que nunca han usado anteriormente una computadora necesitarán leer este libro. También será útil para aquéllos que vienen de otros sistemas, ya que el ORIC tiene muchas características que la hacen más potente que otras máquinas. Si Vd. ya está familiarizado con las computadoras, encontrará fácil pasar al siguiente capítulo.

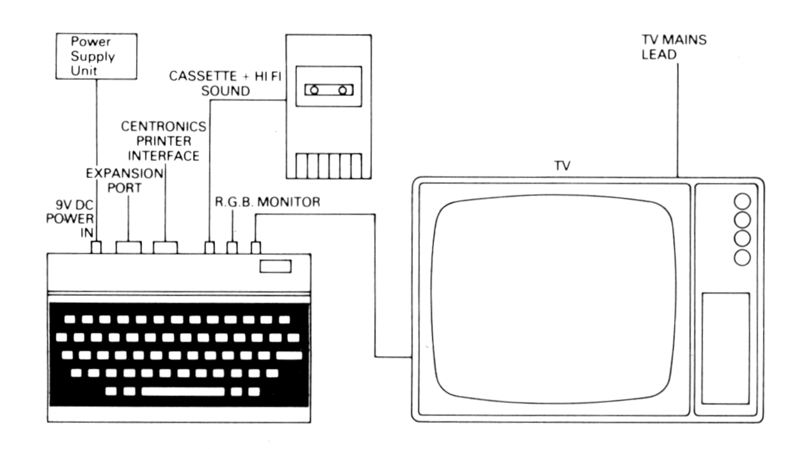
Vd. aprenderá mucho con la lectura de este manual, pero sólo llegará a ser un entendido usando muy frecuentemente el ORIC. Esperamos que Vd. lo encuentre amigable y que pronto tenga todo un gran sistema de computadora basado en el ORIC. Pronto descubrirá la "manejabilidad" de ORIC. Aún para los que empiezan es fácil con ORIC.



CAPITULO 2
Poner en marcha el computador



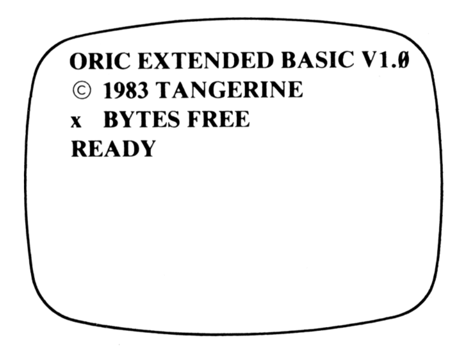
2. Poner en marcha el computador



Cuando Vd. desembale su ORIC, observará que tiene un teclado, para entrar información, y varios enchufes detrás. Necesita ésto para poderse conectar con el mundo exterior.Primero de todo, conecte el alimentador de corriente de acuerdo a las instrucciones que tiene detrás e inserte el enchufe pequeño dentro del zócalo de la parte de atrás. ORIC sólo necesita corriente continua de baja tensión, así pues nunca lo enchufe directamente a la corriente.

El teclado sirve para entrar datos, pero Vd. no verá ningún resultado hasta que no lo conecte a una T.V. Usando el cable de conexión, enchufe un extremo por la parte de atrás del ORIC y el otro al enchufe del U.H.F. de su T.V. La mayoría de las televisiones compradas durante los últimos 15 años funcionarán satisfactoriamente, a pesar de que las televisiones en blanco y negro le darán sombreados en lugar de colores.

Ahora viene el momento que Vd. estaba esperando -enchufe todos los cables, y sintonice el televisor en el Canal 36. Si Vd. tiene un sintonizador rotativo, no será difícil: en caso contrario sintonice con uno de los botones que no utilice para recibir emisoras. Cuando la situación es correcta, aparecerá la siguiente imágen en su T.V.



Si Vd. mira a la parte de atrás del ORIC verá otros enchufes. El diagrama le demuestra sus funciones particulares. Algunas le serán útiles muy pronto, otras se utilizarán cuando Vd. amplíe su sistema.

El enchufe más importante es el del grabador del casette. La mayoría de ellos sirven -pero los portátiles baratos van mejor que los modelos caros en hi-fi. Cuando Vd. escriba un programa y lo quiera salvar, el ORIC convierte el programa en una señal de sonido que se puede grabar. El programa puede ser cargado de nuevo en la máquina cada vez que Vd. lo desee y ¡No necesitará esribirlo de nuevo!.

Vd. necesita un cable con un enchufe DIN de tres o siete patas al final, y un enchufe DIN o un microjack de 3.5mm, de acuerdo a los enchufes de su grabador.

Al lado del enchufe del casette está el del monitor R.G.B. La señal que sale del ORIC, a través del enchufe de la T.V., tiene que ser codificada dentro de una señal del U.H.F. para su aparato de televisión, el cual lo codifica. Un monitor es como un aparato de T.V. sin la seççión de tono ni de sonido. La señal no tiene que ser codificada ni descodificada.

Si Vd. quiere hacer una grabación permanente de lo que imprime el ORIC en la pantalla, Vd. puede conectar una impresora via el siguiente enchufe. Para hacerlo más fácil, la mayoría de los fabricantes hacen impresoras que usan enchufes estandards. El ORIC se conectará a cualquier impresora que tenga una interface Centronics.

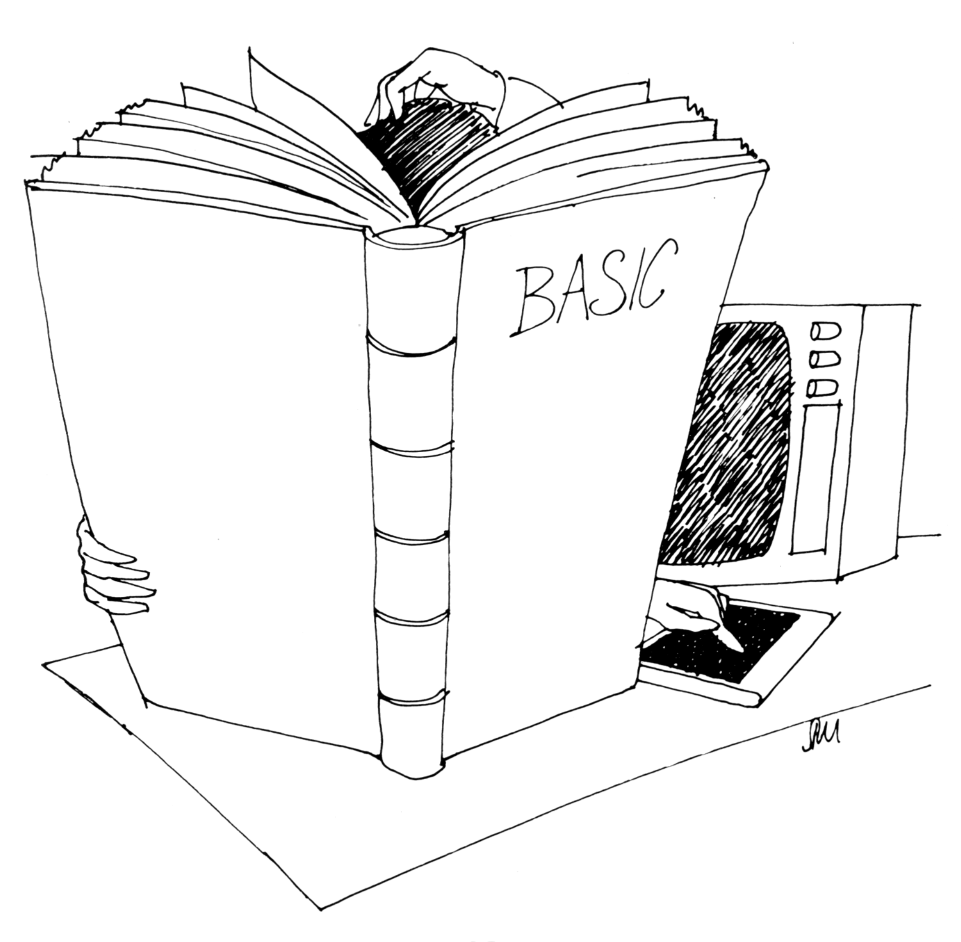
El último enchufe es el más largo, y tiene varias maneras de conectar el ORIC a otras piezas del equipo.

Algunos ejemplos son: memoria extra, cartuchos para juegos, mandos y, naturalmente, el modem. Este periférico le permitirá recibir páginas de TELETEXT e incluso programas; así mismo, enviar y recibir correo electrónico a través del servicio PRESTEL MAILBOX.

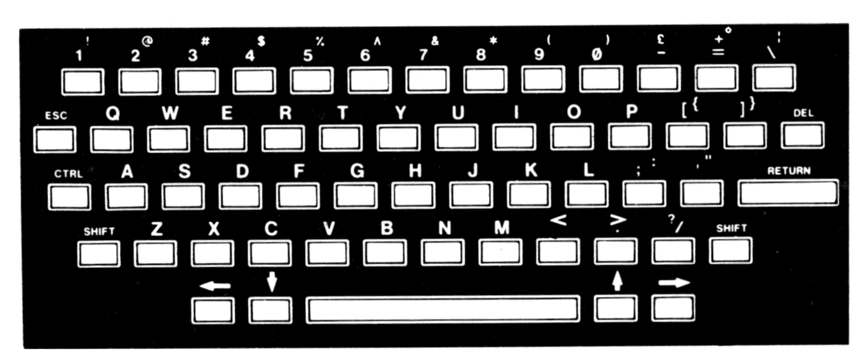
Debajo del ORIC hay un pequeño botón que Vd. necesita un lápiz

para hacerlo funcionar. Es el botón del **RESET** y es un elemento de emergencia para salir de los bucles que nunca terminan. No desconecta la alimentación pero sí para la ejecución de un programa. También se le llama botón de arranque en caliente, ya que no destruye los contenidos de la memoria.

CAPITULO 3
Programación en Basic



3. Programación en BASIC



Primero las malas noticias -el ORIC no entiende Inglés ni español. Pero ahora las buenas noticias -Vd. no tiene que aprender ningún lenguaje electrónico complicado, porque el ORIC habla un lenguaje llamado BASIC, Código de Instrucciones Simbólicas para Uso General del Principiante . Este lenguaje fue inventado en 1964 para ayudar a la gente a escribir más fácilmente los programas de la computadora. Si su máquina está conectada, verá lo fácil que es esto.

Escriba

PRINT"HELLO"



y entonces pulse la tecla **RETURN.**

Las letras aparecerán en la pantalla a medida que Vd. las va escribiendo, cuando Vd. pulsa **RETURN,** la palabra **HELLO** aparecerá debajo. El cuadrado destelleante se llama cursor. Le dice donde aparecen las palabras en la pantalla. **PRINT** es un comando del BASIC, y significa lo mismo que en Inglés (imprimir). Intente

IMPRIMIR algunas otras cosas en la pantalla. Recuerde poner comillas en las cosas que Vd. quiere decir y no olvide pulsar **RETURN.**

Ahora entre

IMPRIME"HELLO"

y pulse **RETURN.**

¡Uf! Vd. acaba de recibir un mensaje de error. Las palabras

"SYNTAX ERROR"

significan que Vd. ha hecho un error. A pesar de que el BASIC es fácil de aprender, y próximo al Inglés, Vd. debe usar las palabras correctas o el ORIC no entenderá nada.

Un vistazo al interior del ORIC le puede ayudar a clarificar un poco las cosas. Dentro del ORIC, hay varios microchips. El más importante es la C.P.U. (Unidad del Procesador Central) que es el cerebro del ORIC. Los microchips usan tensiones altas o bajas para funcionar. Si Vd. se imagina una fila de ocho bombillas con interruptores debajo, entonces puede observar que cualquiera de ellas pueden encenderse o apagarse. Vd. puede pensar de los chips como contenedores de cantidades y cantidades de esos bloques de ocho interruptores. Cada interruptor se llama un bit, y cada bloque un byte.

Si Vd. piensa sobre ello, existen 256 combinaciones diferentes de encendidos y apagados. Si el encendido es = 1 y los apagados son = 0 entonces esto es una manera para almacenar 256 números. Este sistema se llama binario.

Binario		Decimal Normal
00000000	=	0
00000001	=	1
00000010	=	2
00000011	=	3
00000100	=	4
etc.		
11111110	=	254
11111111	=	255

He aquí el por que alguna gente cree que las computadoras sólo están relacionadas con el mundo de las matemáticas -de hecho, los ceros y unos pueden estar en lugar de letras, o palabras, o cualquier cosa. Esto es similar al código morse que puede decir cualquier cosa que quiere usando sólo puntos y rayas.

Otro chip importante del ORIC es la ROM BASIC (Memoria de lectura solamente). Esto traduce las palabras en BASIC a ceros y unos que el cerebro del ORIC puede entender. No hay muchas palabras en

BASIC, un número entre el uno y el doscientos. Hay varias versiones diferentes del BASIC, lo mismo que hay varias versiones del Inglés.

Por ejemplo, el Inglés que se habla en Londres no es el mismo que se habla en Nueva York. Si un Londinés habla del pavimento, un Americano le llamará arcén. Esto es una razón por lo que tenemos que hablar correctamente al ORIC, u obtenderemos un mensaje de error.

Escriba

PRINT "5"

entonces **RETURN**, entonces escriba

PRINT 5

entonces **RETURN** Aparentemente, no hay diferencia.

Ahora escriba

PRINT "5+2"

entonces escriba

PRINT5+2.

(Hasta aquí Vd. ya se ha acostumbrado a pulsar la tecla **RETURN,** así pues pararé de recordarle). Si Vd. entra información encerrada entre comillas, se llama una **string** (cadena alfanumérica), y las strings pueden ser letras, números, ¡y caracteres gráficos!.

Si Vd. entra información sin comillas, entonces el ORIC entiende que es un número, y si está en la forma de una suma, le proporcionará la respuesta.

Intente

PRINT 75+25

Debería obtener un **100.** Sugerencia -Vd. puede escribir **?** en lugar de **PRINT** para ganar tiempo. (observe la diferencia entre un 0 y una mayúscula O, y entre un 1 y la mayúscula I- Vd. sabe lo que significa, pero al ORIC se le ha de especificar correctamente).

Intente otros cálculos. La tecla de la resta está situada al lado del 0, la de dividir es **/** y la de multiplicar es ***** (shift 8).

Si Vd. comete un error, pulse **CTRL** y **X** simultáneamente. Aparecerá una barra y se cancela la línea entera.

Si Vd. está interesado en matemáticas más complicadas, recuerde que el ORIC no opera en números tal como llegan.

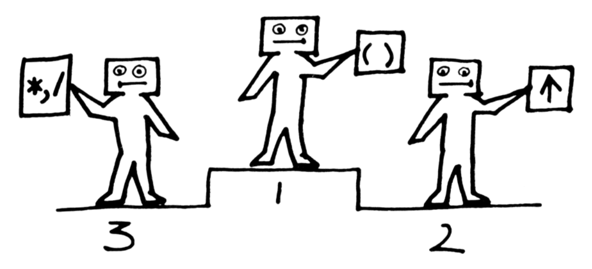
PRINT 4+3*2 no da como resultado **14**, pero sí **10** porque * es más importante que el +. Aquí existe el orden de prioridad, la primera es la más importante y la última la que menos.

()

↑ ("a la potencia de")

***,/**

+,-



Los operadores en la misma línea tienen la misma prioridad. Esta es la manera de usar el ORIC como una calculadora.
Entre estos ejemplos para entender estas prioridades.

```
PRINT 2*3*4
PRINT 4+3*2
PRINT 4*3+2
PRINT 4/2+3
PRINT 2+3/4
PRINT 3+4↑2
PRINT 3-4↑2
PRINT 2+4↑3*2
```

Si Vd. no está del todo seguro entonces ponga primero la parte del argumento (ecuación) que Vd. quiere calcular entre corchetes, o sea:

4+3*2=10

pero

(4+3)*2=14

* * * * * * * * *

¿Recuerda que le dije que el ORIC entiende que todo lo que va sin comillas es un número?

Escriba
PRINT H

y vea lo que sucede, obtiene un 0. Naturalmente no puede ser un número -¿o sí?. Escriba

LET H= 4.

Ahora escriba
PRINT H.

Esta vez el ORIC sabe que Vd. ha puesto que **H** es igual a **4** (lo mismo que en álgebra). A la **H** se le llama una **variable.** El ORIC recordará esto hasta que Vd. cambie el valor de **H,** o escriba **CLEAR,** o lo desconecte. Inténtelo con otras letras, entonces escriba **CLEAR** y vea si el ORIC se ha olvidado. Vd. puede usar más de una letra, de manera que **AB** puede tener un valor diferente a **A** o **B.** Vd. también puede tener **A5** o **A6.** El ORIC aceptará las variables de más de dos caracteres de longitud, pero sólo reconocerá las dos primeras.
Intente:-
LET JOHN = 36
LET JOCELYN = 28

Ahora escriba

PRINT JOHN
PRINT JOCELYN

Vd. debería obtener cada vez un **28,** porque el ORIC sólo ha recordado una variable **JO,** que a pesar de ser determinada a **36,** se le cambió su valor por **28.**
Entre
PRINT 4*JO

y Vd. puede observar como las variables se usan como simples números. Hasta ahora sólo hemos discutido sobre números. ¿Cómo puede el ORIC recordar los nombres? Lo mismo que hay números y variables también, hay strings y variables string (cadenas alfanuméricas).
Escriba
LET N$ = "BLAKE"
Ahora escriba
PRINT N$.

Vd. debería ver que **"BLAKE"** ha sido recordado como una variable de string. Intente determinar **FP$** a su nombre preferido de persona. Si Vd. escribe

PRINT N$, FP$

Vd. debe obtener ambos nombres en la pantalla. La coma fija el espacio entre las strings, como usando **TAB** en una máquina de escribir. La ORIC tiene fijadas las tabulaciones a cinco caracteres. Si Vd. usa un punto y coma **(;)** no hay ningún espacio entre las variables, o sea

PRINT N$;FP$

Es posible sumar strings para hacer una string nueva.

Escriba

LET A$= N$ + FP$

entonces **PRINT A$.** A esto se le llama **concatenación.** Intente usar otra vez **CLEAR** para repasar si borra también las **variables de la string.**

Nota: -Vd. no tiene que usar el **LET** para asignar valores a las variables. Ejemplo: **A=10** es lo mismo que **LET A=10** lo es para la ORIC.

Cuando hablamos de las **variables de número,** Vd. probablemente descubrió que las fracciones decimales podrían estar en las variables, lo mismo que los números.

Intente:

LETX = 1/3

PRINT X

Vd. debe obtener **0.33333333**

(El ORIC puede contener números entre 2.93874×10^{-39} hasta 1.70141×10^{38}). Para más información, vea el Capítulo 6.

Las letras simples se denominan **variables de coma flotante.** Si la letra, o par de letras, va seguida por un **%** entonces se llama **entero** o variable entera, ej.: **A%=4762.** Puede estar entre -32768 y +32767. En general, las variables enteras se manejan más rápidamente que las variables de coma flotante.

Hasta aquí, hemos usado el ORIC para que nos de los resultados de una forma muy simple. A esto se le llama **ejecución inmediata** o **modo de calculadora.** La manera más normal de usar las computadoras es hacerles almacenar una serie de instrucciones y usar este programa cuando se necesite.

En BASIC, el orden de las acciones está controlado por números de líneas. Es normal fijar los números de línea de 10 en 10 para así poder insertar más tarde nuevas líneas. No importa en que orden entre las líneas. El ORIC las colocará automáticamente en orden. Pruebe este programa corto.

```
10    CLS
20    PRINT "TECLEE SU NOMBRE"
30    INPUT N$
40    PRINT "ENCANTADO DE CONOCERLE,";N$
```

El ORIC irá al número de la primera línea, 10 y borra la pantalla, **PRINT** "Teclee su nombre" y entonces va a la línea 30. Esta dice **INPUT N$**, de manera que el programa se parará aquí hasta que Vd. entre algo y escriba **RETURN. N$** contiene su nombre. El ORIC pasará a la línea **40** e imprimirá

"ENCANTADO DE CONOCERLE"

seguido de su nombre. El punto y coma **(;)** imprime su nombre a continuación. Los puntos y comas no son necesarios en las sentencias **PRINT** pero dejan el listado más claro. Omiten el avance de línea, así pues tenga cuidado al colocarlos al final de líneas. Escriba **RUN.** Este ejecuta el programa desde el número de línea más pequeño -también borra cualquier variable anteriormente fijada, de manera que Vd. puede dejar ejecutando el programa con nombres diferentes.

Aunque no es necesario con el ORIC, es normal poner **END** (FIN) como última línea del programa.

Escriba **LIST** y todo su programa aparecerá en la pantalla. Ahora añada estas líneas.

```
 50    PRINT "TECLEE EL AÑO EN QUE NACIO VD."
 60    INPUT YEAR
 70    LET AGE= 1983-YEAR
110    PRINT "VD. DEBE TENER ALREDEDOR DE";AGE;N$
120    GOTO 200
200    END
```

Si Vd. se equivoca en una línea Vd. puede borrar la línea entera entrando simplemente el número de esa línea. Pruebe de escribir **60** entonces **LIST**e el programa. Reescriba **60 INPUT YEAR.** Ahora EJECUTE **(RUN)** el programa.

* * * * * * * * *

DECISIONES

Hasta aquí hemos utilizado las computadoras para trabajar con todos los números sin tomar ninguna decisión. Usemos el cerebro del ORIC un poco más. Añada estas líneas. (¡Las feministas deben desear cambiar las palabras de las líneas 80 + 150!)

```
 80   PRINT "PERDONE POR LA PREGUNTA, PERO ¿ES VD.
      HEMBRA";N$;" (S/N)?"
 90   INPUT A$
100   IF A$="S" THEN 150
150   PRINT "BUENO, "N$" UNA CHICA TAN ATRACTIVA
      COMO TU DEBE TENER UNOS 18 AÑOS"
```

La línea 100 contiene una bifurcación condicional. **A$** puede ser **"S"** o **"N"**. El ORIC verifica si **A$** = **"S"**. Si esta aserción es verdadera ENTONCES **(THEN)** el programa salta a la línea 150. Si esta aserción es falsa, o sea si nada más ha sido entrado, entonces el programa continua a la línea siguiente, imprime el año, y se para. Vea el final de este capítulo para mayor información sobre **ELSE**.

Probablemente verá que el programa sólo verifica la **"S"** como respuesta. **"OK"** o **"VALE"** o **"ESO ES"** no se cuentan como **"S"** y por lo tanto como falso -así pues tenga cuidado, aunque el ORIC es muy bueno obedeciendo instrucciones, deben de especificarse en primer lugar muy cuidadosamente.

Puede probar de alterar la línea 100 para poder con otras respuestas. Puede empezar con

```
100 IF A$ = "S" OR A$ = "SI" O A$ = "EA"
    THEN 150
```

Hasta ahora todo lo que Vd. ha estado imprimiendo ha sido en mayúsculas. Quizás ha descubierto que el **SHIFT** parece que no funciona.

Si Vd. pulsa **CTRL** y **T** a la vez, encontrará que el teclado del ORIC actúa como una máquina de escribir -letras en minúscula (letras pequeñas) y letras en mayúscula cuando Vd. pulsa al mismo tiempo la tecla **SHIFT**. Si Vd. pulsa control **T** (**CTRL** y **T**) de nuevo, el ORIC sólo escribirá en mayúsculas otra vez.

Para que Vd. sepa si está en modo CAPS (MAYUSCULAS), el ORIC imprime la palabra CAPS en la línea de arriba de su pantalla.

Esto tiene mucha importancia. Si Vd. **ejecuta** el programa en letras minúsculas, obtendrá

? SYNTAX ERROR.

Todos los comandos y variables en BASIC deben de escribirse en mayúsculas.

Vd. puede escribir todo lo que quiera en el ORIC y pulsar cualquier tecla sin por ello dañar a la computadora. Lo peor que puede suceder es que se entre en un bucle sin final o se le quede la pantalla "sucia", -si ve que algo extraño le sucede a la pantalla, primero pulse **RESET.** Si no sucede nada, desenchufe todos los elementos, espere durante unos sgundos y vuélvalos a encender. Obtendrá otra vez la pantalla original. El ORIC no se quejará, ¡pero quizás Vd. tenga que volver a reescribir su programa!.

Entonces ¿qué ocurre cuando Vd. ya está cansado de un mismo programa y quiere comenzar otro?. En lugar de desenchufarlo todo, Vd. sólo tiene que escribir **NEW** (NUEVO). Esto borrará la memoria y fijará todas las variables a cero.

* * * * * * * * * *

BUCLES

El ORIC nos ha demostrado hasta ahora que las computadoras son capaces de tomar decisiones de si las condiciones son falsas o verdaderas. También son capaces de repetir una acción cuantas veces Vd. necesite.

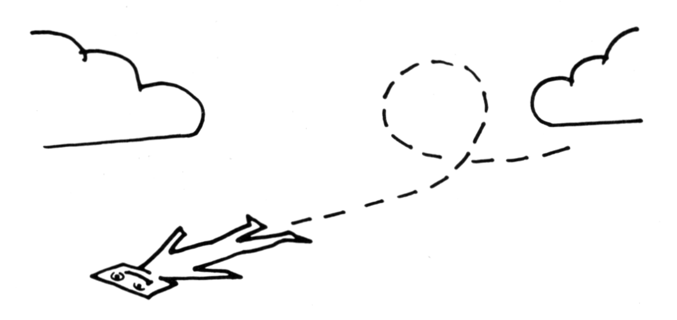
Por ejemplo, si Vd. quiere que el ORIC imprima todos los números del 1 al 1000 y que salgan todos por la pantalla, escriba:-

```
10    PRINT 1
20    PRINT 2
30    PRINT 3
40    PRINT 4
```

¡Pero quizás llegue a cansarse de hacer esto 1000 veces!. Por suerte hay un comando en BASIC llamado **FOR,,,TO/NEXT**, que repetirá la instrucción hasta que llegue al número final.

Así es como funciona:-

```
10    FOR X = 1 TO 1000 STEP 1
20    PRINT     X
30    NEXT X
40 PRINT "BUF!!. HE TERMINADO"
```



La línea 10 determina el contador **X** a 1, entonces va a la línea 20. Allí imprime **X**, que es 1, y va a la 30. Dice **NEXT X**, de manera que vuelve al bucle de la línea 10 y la incrementa con el número **STEP** (o sea, **X** es ahora 2). Imprime **X** en la línea 20 entonces se repite hasta que **X** es 1000. Esta vez intenta hacer que la **X** = 1001, pero se le ha dicho que vaya hasta la 1000, de manera que salta a la siguiente línea -40, donde imprime

"HE **TERMINADO"**.

Si Vd. cambia el número del **STEP** en 2, entonces imprimirá **1,3,5,7** etc. Si no pone **STEP** el ORIC entenderá que Vd. quiere que la medida del **STEP** sea 1.

Pruebe Vd. mismo otras mediadas para el **STEP.**

Los bucles **FOR/NEXT** pueden contar hacia atrás, pero Vd. tiene que especificar el número del **STEP** como una cantidad negativa.

```
10    FOR X = 1000 TO 1 STEP -1
```

contará hacia atrás.

Si Vd. comete un error en los números, o sea

```
FOR X = 4 TO 2 STEP 5
```

6

```
FOR X = 5 TO 100 STEP -1
```

entonces la acción en el bucle se volverá a realizar por lo menos una vez, porque la verificación de donde termina el bucle no se hace hasta que éste llega a la sentencia **NEXT** y vuelve al comienzo.

Otra función de los bucles **FOR/NEXT** es el de una pausa. Probablemente encuentre que el ORIC imprime tan deprisa que Vd. no puede ni ver los números. Para ir más despacio, puede poner

```
25      FOR PAUSA = 1 TO 10: NEXT PAUSA
```

Esto es como decir "Cuente hasta diez cada vez que imprima un número, entonces continue".

Tenga cuidado cuando use esta función si hay un ramificación **IF...THEN** en la línea, como cuando la condición es verdadera que el ORIC entonces salta a la línea nueva e ignora cualquier otra sentencia en la línea original.

Una manera más fácil para conseguir las pausas en el ORIC es usando el comando **WAIT** (ESPERAR).

25 WAIT N parará la ejecución del programa para N cantidades de 10 milisegundos.

* * * * * * * * *

SUBRUTINAS

Vd. puede estar pensando acerca de las secciones de un programa que ocurren varias veces, pero que no las puede realizar usando simples bucles **FOR/NEXT**.

Por ejemplo, en nuestro programa de contar, Vd. puede querer decirle a la gente que la espera entre los números fue intencionada.

Vd. envía el programa a una **subrutina**, donde espera, imprime un mensaje, entonces retorna al lugar donde dejó el programa principal.

```
  10    FOR X = 1 TO 10
  20    GOSUB 1000
  30    PRINT X
  40    NEXT
1000    PRINT "ESTO ES UNA INTERRUPCION CORTA"
1010    WAIT 50
1020    RETURN
```

* * * * * * * * *

ON...GOTO

Algunas veces, en el curso de un programa, es útil poder ramificar a diferentes partes del programa de acuerdo a los resultados de

algún cálculo. Esto es fácil usando el comando **ON...GOTO.** Todo lo que Vd. necesita saber son los resultados de los cálculos y los números relevantes de las líneas para ramificar.

```
 50   INPUT "ELIJA ENTRE 1, 2 ó 3";X
 60   ON X GOTO 100,200,300
 70   PRINT "ELECCION ERRONEA": STOP
100   PRINT "ELIGIO 1": STOP
200   PRINT "ELIGIO 2": STOP
300   PRINT "ELIGIO 3": STOP
```

La línea 50 espera una entrada. Si X es 1, el control se ramifica al número de la primera línea después del **GOTO,** o sea 100; si X es 2, se ramifica al segundo número de línea, o sea 200 y si es 3, se va al tercer número, o sea 300. Si se entra otro número, el programa continua hasta la siguiente sentencia **ON...GOTO**

Un comando parecido es el **ON...GOSUB,** el cual se ramifica a una subrutina determinada. Cuando el programa retorna, continuará desde la siguiente sentencia después del **ON...GOSUB.**

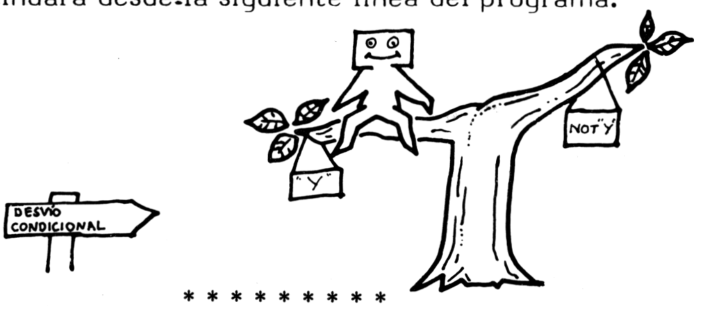
* * * * * * * * * *

¿QUE MAS?

Hasta aquí sólo hemos usado **IF/THEN** en su forma simple. Es posible extender su potencia usando **ELSE.** Observe esto:

```
10   FOR X = 1 TO 5
20   INPUT A
30   IF A = 10 THEN PRINT "DEMASIADO GRANDE" ELSE
     PRINT "O.K."
```

Si la condición es verdadera entonces se ejecuta el primer comando; si es falsa, entonces el comando que sigue a **ELSE.** Si al programa no se le ha dicho que se ramifique, entonces la ejecución continuará desde la siguiente línea del programa.



* * * * * * * * *

REPEAT/UNTIL

Si Vd. desea repetir una serie de instrucciones un cierto número de veces, entonces es fácil usar un bucle **FOR/NEXT.** Se repetirán tantas veces como lo indique la primera línea, por ejemplo:

FOR N = 1 TO 5

se ejecutará el bucle cinco veces. Si Vd. desea hacer un bucle hasta encontrar que una condición es verdadera, es difícil conocer el valor que se ha de poner en el contador del bucle.

REPEAT (REPETIR) le permite hacer un bucle las veces que quiera, y verificar el final de cada bucle para revisar si la condición se cumple en la línea **UNTIL** (HASTA). Este programa corto demuestra esto

```
10    REPEAT
20    D= D + INT(RND(1)*6) + 1
30    PRINT D
40    UNTIL D> 20
50    STOP
```

Esto simula una situación que durará mientras se esté lanzando un dado, y continuará hasta que el total lanzado exceda a 20 veces. No será posible conocer el número de bucles antes de la condición en la línea 40, por eso no se puede usar un bucle **FOR/NEXT.** Sería posible imitar esta acción usando una sentencia **GOTO SI** la condición no se encontrara, pero la estructura del programa no sería clara al leer el listado, por eso **REPEAT** debe utilizarse donde sea posible.

Observe que lo mismo que con **FOR/NEXT**, la condición se verifica al final, de manera que el bucle siempre se ejecuta por lo menos una vez.

* * * * * * * * *

Antes de entrar en áreas más interesantes como los dibujos y la música, queda por hacer que Vd. haga sus programas más fáciles de leer.

Use las sentencias **REM** para explicar las líneas. **REM** sirve para hacer anotaciones y es ignorada por el ORIC. Su aplicación es útil para cuando Vd. lee un listado o cuando Vd. muestra el programa a otros. Esto le demostrará como puede Vd. utilizar **REM.**

```
10    REM PROGRAMA DE PEPE
20    FOR N = 1 TO 10: REM BUCLE CONTADOR
30    PRINT "PEPE PROGRAMA BIEN"
40    NEXT N
```

```
50    END
60    REM ESTE ES UN PROGRAMA MAS BIEN BOBO
```

Use las sentencias **REM** para etiquetar sus subrutinas. Observe como Vd. puede haber tenido más de una sentencia en una línea, pero cada sentencia puede estar separada por dos puntos -Vea la línea 20.

```
1000    REM SUBRUTINA DE ESPERAR UN RATITO
1010    WAIT 100
1020    RETURN
```

Vd. puede usar ' en lugar de **REM,** pero sólo como una observación al final de una línea.

```
10    PRINT "HOLA" 'DECIR HOLA
```

es admisible.

```
10    'MARCA REGISTRADA DE ORIC LTD
```

no es admisible.

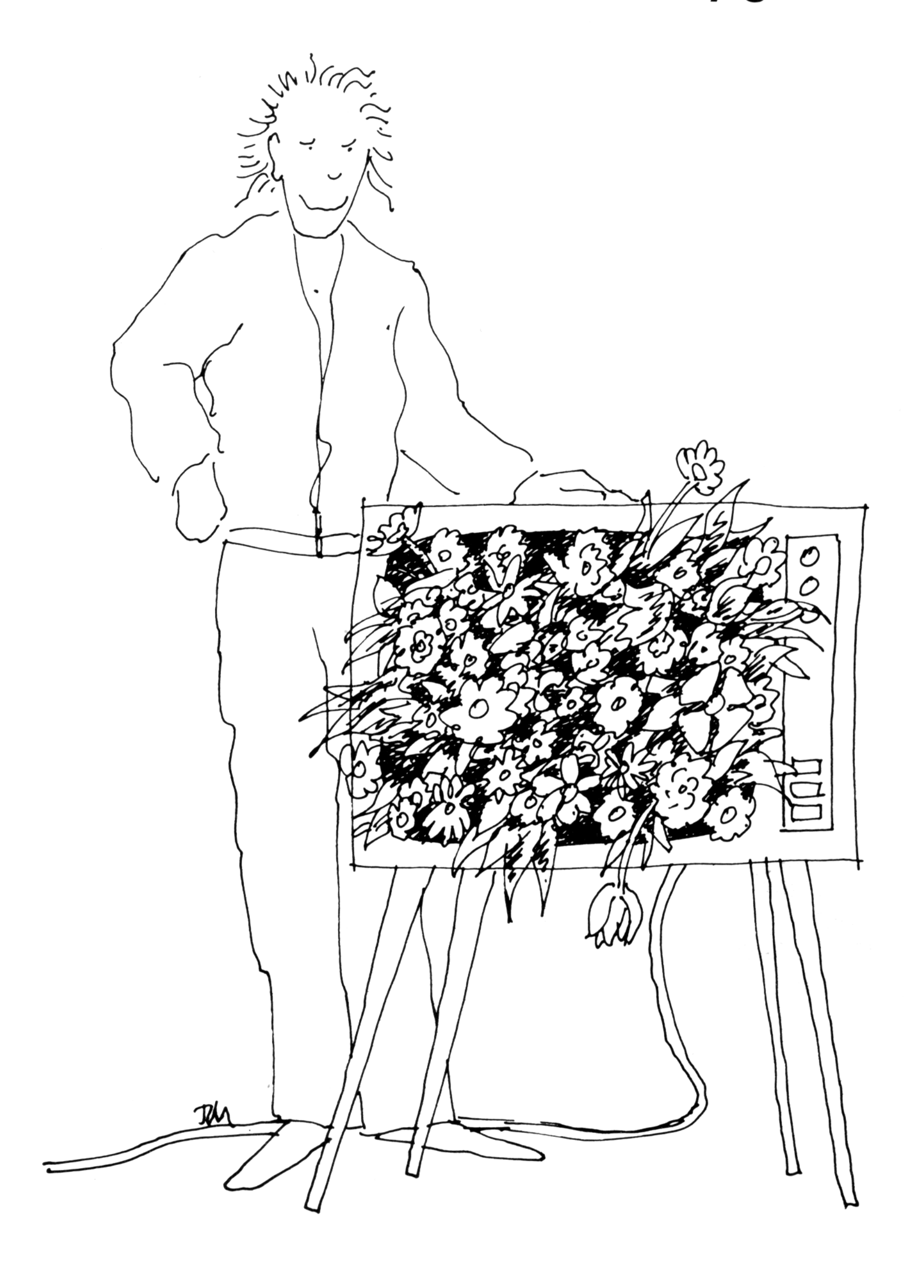
El Basic no es difícil de aprender, y esto es sólo una breve guia. Vd. se tornará más eficiente cuanto más lo practique.

* * * * * * * * * *

oigo y olvido
veo y recuerdo
hago y entiendo

Viejo Proverbio Chino.

CAPITULO 4
Color y gráficos



4. Color y gráficos

Cuando Vd. conecta el ORIC entra automáticamente en modo **TEXT**, o sea Vd. puede usar la pantalla para escribir en ella directamente y cuando está llena, las líneas irán subiendo automáticamente. La área del texto también se usa para gráficos de baja resolución.

Antes de que Vd. practique con **LORES**, sería una buena idea descubrir los colores que se pueden usar. Hay dos comados para el color, **INK** y **PAPER**. Esto determina los colores de la pantalla y los del fondo, respectivamente, y pueden utilizarse tanto en comandos directos como en programas. Deben ir seguidos por un número (0 al 7) para especificar el color, y pueden usarse en los modos **TEXT** o **HIRES**.

```
0    NEGRO
1    ROJO
2    VERDE
3    AMARILLO
4    AZUL
5    MAGENTA
6    CIAN
7    BLANCO
```

Ahora intente con alguno de ellos. Si Vd. ya está familiarizado con las computadoras en las que hay que borrar la pantalla antes de poder cambiar el color, verá que el ORIC no lo necesita.

A continuación verá un programa corto pra demostrar todas las combinaciones de colores de su ORIC.

```
 5    REM COLORES
10    TEXT
20    FOR N = 1 to 25
30    PRINT "ESTE TEXTO ESTA EN EL COLOR INICIAL"
40    NEXT N
50    FOR I = 0 TO 7
60    FOR P = 0 TO 7
70    INK I = PAPER P
```

```
 80   WAIT 100
 90   NEXT P
100   NEXT I
210   INK 7: PAPER 4
```

Naturalmente cuando los colores de las letras y del fondo son iguales, ¡no podrá leer las palabras!



Para gráficos de baja resolución, Vd. puede usar la pantalla en modo **TEXT** o entrar **LORES 0** o **LORES 1.** La pantalla disponible para dibujos es de 0 a 38 en el eje X (horizontal) y de 0 a 26 en el eje Y (vertical). La posición 0,0 está en el extremo izquierdo de arriba de la pantalla. La columna de más a la izquierda no se puede usar, porque contiene el atributo que controla el color del fondo o **PAPER** de una fila.

La siguiente columna controla el color de los caracteres o **INK**, pero puede usarse en modo **TEXT.** Si **LORES 0** o **LORES 1** son seleccionados, la pantalla queda con fondo negro, y el atributo del conjunto de caracteres alternativos también se coloca a la izquierda de la pantalla.

LORES 0 usa el juego de caracteres estandard, y **LORES 1** usa el juego de caracteres alterno.

Pruebe esto:

```
10   LORES 0
20   PLOT 16,12, "HOLA"
```

Si Vd. ejecuta este programa corto, se imprimirá **HOLA** en el centro de la pantalla. Vd. podría usar el programa del Capítulo 9 para definir otros caracteres usando el conjunto estandard. El comando **PLOT** le ahorrará hacer un **POKE** en la memoria de la pantalla.

Ahora si Vd. escribe

```
10    LORES 1
```

Si la línea 20 está todavía intacta, en lugar de aparecer **HOLA**, se imprimirá un extraño número de bloques. Son caracteres del grupo alterno, y estos en particular son los que tienen los mismos códigos de ASCII que las letras **HELLO.** Esta es la única diferencia entre **LORES 0** y **LORES 1.**

Este programa imprimirá completamente el grupo de caracteres alternos.

```
 5    REM ** JUEGO DE CARACTERES ALTERNO **
10    FOR N = 32 TO 128
20    PRINT N, CHR$(27); "I"; CHR$(N)
30    PRINT
40    WAIT 25
50    NEXT N
```

Vd. puede utilizar este programa para seleccionar caracteres para formar sus propias figuras de gráficos. Esta es una manera en que puede utilizarse.

```
 1    REM *** MONSTRUO ***
 2    REM *** DEMO LORES 0/1 ***
 5    LORES 1
 6    D = 0
 9    REPEAT
10    A$ = "F9":B$="6I"
20    FOR C= 0 TO 35
30    PLOT C,D,A$
35    PLOT C,D+1,B$
45    PLOT C,D," "
50    PLOT C,D+1," "
55    NEXT C
56    SHOOT
60    D=D + 2
70    UNTIL D=26
75    EXPLODE
80    CLS
```

Los caracteres en **A$** y **B$** no aparecen en su forma normal, tal como **LORES 1** ha sido seleccionado. El resto del programa envía el caracter elegido sucesivamente a través de las hileras hasta que se llega a la fila 26, ¡cuando explotará!.

Para ver los caracteres estandard, simplemente escriba **LORES 0** en la línea 5. Si Vd. necesita mezclar los caracteres alternos y estandard en la misma pantalla, o sea mezclar texto y gráficos, entonces es fácil. Para usar caracteres estandard en **LORES 1, CHR$(8)** le pondrá en estandard, y **CHR$(9)** volverá al modo anterior.

Si Vd. los usa en inverso, naturalmente Vd. puede imprimir los caracteres en **LORES 0.** A continuación verá dos programas para demostrarlo. En todos los programas que usan **LORES 0** o **LORES 1,** es una buena idea desconectar el cursor destelleante pulsando **CTRL** y **Q** al mismo tiempo. Repitiendo esta acción retornará el cursor otra vez.

```
 5    REM ** TEXTO EN LORES 1 **
10    LORES 1
20    A$= CHR$(8) + "HOLA" + CHR$(9)
30    FOR N= 2 TO 24
40    PLOT N,N,"KKKK"
50    PLOT N,26-N, A$
60    NEXT N
70    WAIT500
80    CLS
```

```
 5    REM ** TEXTO EN LORES 0 **
10    LORES 0
20    A$=CHR$(9)+"HOLA" +CHR$(8)
30    FOR N = 2 TO 24
40    PLOT N,N,"KKKK"
50    PLOT N,26-N,A$
60    NEXT N
70    WAIT 500
80    CLS
```

* * * * * * * * *

POSICIONES DE LA PANTALLA

Si Vd. necesita conocer que caracteres están en una posición determinada de la pantalla en los modos **TEXT** o **LORES,** use **SCRN(X,Y).**

Escriba **CLS** para borrar la pantalla. El cursor estará en el extremo superior izquierdo de la pantalla. Escriba

PLOT 10,20,"A".

Aparecerá una **A** en mayúscula cerca del principio de la pantalla.

Escriba
PRINT SCRN(10,20)

El número **65** será devuelto, ya que es el código ASCII para la letra A.

A continuación verá un programa corto que **REPITE** un bucle hasta que un misil alcanza su objetivo. **SCRN(X,Y)** detecta cuando el misil está a una posición del objetivo (el código Ascii para + es 43 -ver línea 220), y el programa termina con una explosión. Después de haber ejecutado el programa, cambie el modo en el cual trabaja, añadiendo la línea **115 LORES 0** o **115 LORES 1.** Esto le demostrará los diferentes efectos que Vd. obtiene de acuerdo al modo que Vd. ha elegido.

```
100    :REM ** USO DE SCRN(X,Y) **
110    :CLS:INK1:PAPER4
120    :FOR N= 20 TO 25
130    :      PLOT N,26,"+"
140    :NEXT N
150    REPEAT
160    :      A=INT(RND(1)*36+2)
170    :      FOR P= 0 TO 24
180    :      PLOT A,P,"V"
190    :      WAIT 4
200    :      PLOT A,O," "
210    :      NEXT P
220    :UNTIL SCRN(A,P+1)=43
230    :EXPLODE
```

Este programa ha intentado hacerlo más fácil para entenderlo. Funcionará perfectamente sin los dos puntos y y espacios. Para más detalles, consulte el Capítulo 12.

Para completar esta sección en gráficos de baja resolución, verá a continuación un programa que le demuestra como con los colores del fondo de la pantalla pueden dibujarse en un círculo.

```
10    REM **** GRAFICOS COLOR EN MODO LORES ****
20    LORES 0
30    STP= 2*PI/50
40    R= 10: X=10:Y =10
50    REPEAT
60    E = 18 + RND(1)*6
```

```
 70     PLOT X+R*SIN(C),Y+R*COS(C),E
 80     C= C+STP
 90     UNTIL C> 2*PI
100     REPEAT: UNTIL KEY$ <> ""
110     CLS
```

* * * * * * * * * *

GRAFICOS EN ALTA RESOLUCION

Si Vd. quiere dibujar gráficos en alta resolución, necestia escribir **HIRES.** Pruébelo ahora.

Observe como el principio de la pantalla se convierte en negro, dejando tres líneas al final para el texto. Esto es útil porque Vd. puede escribir las instrucciones para los dibujos, y ver el efecto en la parte superior de la pantalla. En código inmediato, Vd. puede usar el ORIC como una lámina de dibujo para probar sus instrucciones. Cuando están correctas, Vd. las puede incorporar en sus programas.

Si Vd. quiere retornar a la pantalla de texto, escriba **TEXT**, y la pantalla volverá a su formato original. **TEXT** y **HIRES** pueden utilizarse como comandos en los programas.

Antes de empezar a dibujar algo, tiene que imaginar que la pantalla está dividida en 240 posiciones (etiquetadas 0-239) a través de la pantalla, y 200 posiciones, (etiquetadas 0-199) en vertical. Las que cruzan se llaman posiciones X, y las que bajan son las posiciones Y. Si Vd. ha utilizado gráficos, entonces esto le será familiar -la única diferencia es que el origen (0,0) está en el extremo superior izquierdo de la pantalla.

Hay varios comandos especialistas en dibujos que hacen que los gráficos en el ORIC sean fáciles.

Vamos a verlo para que Vd. pueda ver sus efectos.

CURSET fija el cursor a una posición absoluta **X,Y,** o dibujará ese punto. Debe ir seguido por tres parámetros. (Son números que el ORIC debe conocer).

El cursor **HIRES** no destellea en la pantalla como el cursor de **TEXT.**

ejemplo:-

CURSET 120,100,1

llevará el cursor al centro de la pantalla e imprimirá un punto pequeño. El primer parámetro es cuan lejos a través de la pantalla (0-239), el segundo es cuan lejos verticalmente la pantalla (0-199) y el tercero es el número **FB** (FOREGROUND (PRIMER TERMINO)/BACKGROUND (FONDO)).

Los códigos de **FB** son:-

0	color del fondo
1,	color del primer término
2,	colores invertidos
3,	nulo (no hace nada)

Ahora escriba **HIRES** y practique con **CURSET.**

El siguiente comando de gráficos es **CURMOV.** Es similar a **CURSET** excepto que **X&Y** son relativos a la posición del cursor. Otra vez, el ORIC necesita saber los números de **X, Y,** y de **FB.** Asegúrese de que el valor de **X** o **Y** más la actual posición del cursor no le sacan de la pantalla, pues obtendrá un mensaje de error.

DRAW X, Y, FB dibujarán una línea recta desde la actual posición del cursor al actual cursor más X e Y.

Pruebe este programa corto. Debe encontrar que dibuja un cuadrado. Tenga en cuenta que los números negativos dibujan de derecha a izquierda o de abajo a arriba. Si la forma no es lo suficientemente "cuadrada", pruebe cambiando las líneas 30 y 50.

```
 5    REM ** CUADRADO **
10    HIRES
20    CURSET 60,40,3
30    DRAW 120,0,1
40    DRAW    0,120,1
50    DRAW -120,0,1
60    DRAW 0, -120,1
```

RECUERDE: -Cambiando los modos, o escribiendo otra vez **HIRES**, ¡borrará su dibujo para siempre!

* * * * * * * * * *

MASCARAS (PATTERN)

El ORIC todavía tiene para Vd. otro truco. Cuando Vd. lo enchufa, el comando **DRAW** se fija para dibujar una línea continua. Sin embargo, es posible dibujar líneas de puntos, de rayas, etc., de acuerdo a su propia especificación.

Así es como funciona. Si Vd. recuerda, anteriormente fue mencionado como el ORIC piensa en bytes de 8 bits, de manera que Vd. para contar del 0 al 255 está usando 8 ceros y unos. Cuando se conecta el ORIC, el número 255 se carga dentro de una máscara. (255 se escribe en código binario como 1111111).

Vd. puede fijar la máscara a cualquier número desde el 0 al 255 para conseguir diferentes resultados. Si Vd. desea rayas de igual medida, escriba **PATTERN 15.**
15 es en binario 00001111

8+4+2+1, o sea media línea es "1" y la otra media es "0". Intente dibujar el cuadrado otra vez, pero esta vez cambie el valor de la máscara a números diferentes. No hay nada que pueda parar el que Vd. tenga continuamente dos partes, una con puntos y otra con rayas. Para ayudarle a entender como funciona, añada esta línea al principio.

15 PATTERN 170
(170 es 10101010 en binario).
¡Debería enviarle todo de puntos!

* * * * * * * * *

Para ver realmente las extraordinarias capacidades del ORIC en modo **HIRE**, pruebe este programa corto que genera interferencias dibujando líneas unas junto a otras.

```
 5   REM ** MOIRE **
10   HIRES
20   FOR A= 0 TO 1
30   FOR B= o TO 239 STEP6
40   CURSET 0, 199*A,3
50   DRAW B,199-398*A,1
60   CURSET 239,199*A,3
70   DRAW -B,199-398*A,1
80   NEXTB:NEXTA
```

* * * * * * * * *

CHAR

Si tuviera que intentar **PRINT** (IMPRIMIR) en modo **HIRES,** sólo conseguiría texto en las últimas tres líneas. Sin embargo, hay un comando que le permite imprimir donde Vd. quiera en la pantalla de alta resolución. Escriba **NEW** para borrar la memoria, después **HIRES.** Ahora coloque el cursor a la mitad de la pantalla escribiendo **CURSET 120, 100, 3.** Ahora escriba **CHAR 65,0,1.** Obtiene una **A** en mayúscula en la mitad de la pantalla.

CHAR es el comando, y le siguen tres parámetros.

CHAR X, S, FB.
X es el código A.S.C.I.I. (32-127)
S es 0 (caracter estandard) ó 1 (serie alternativa)
FB es el valor (0-3) del primer término/fondo de la pantalla.

A.S.C.I.I.(generalmente pronunciado "Aski") significa Código Estandard Americano para Intercambio de Información. Está completamente estandarizado y asigna números del código a las letras, figuras y símbolos. (Vea el apéndice). Hay un comando en BASIC, ASC, que devuelve el valor de un caracter en un string, y ésto le ahorrará tiempo tal como verá en el siguiente programa.

```
 5   REM
10   HIRES
20   CURSET 50,50,3
30   N$="HOLA,SOY EL ORIC"
40   FOR A=1 TO LEN(N$)
50   CHAR ASC (MID$(N$,A,1),0,1
60   CURMOV 10,10,0
70   NEXTA
```

Las líneas 40 a la 70 contienen un bucle que mide la longitud de **N$** e imprime los caracteres de acuerdo al comando **CURMOV.**

Cambie **N$** a su nombre, o pruebe cambiar los parámetros de **CURMOV** de manera que Vd. vea lo que sucede. Tenga cuidado de no salirse de la pantalla, u obtendrá un mensaje de error.

* * * * * * * * * *

CIRCULOS

Para dibujar círculos, simplemente escriba **CIRCLE,** seguido de dos números -primero el radio, después del código de **FB.** El centro estará en la actual posición del cursor. Asegúrese de que el radio no saque la circunferencia fuera de la pantalla. Pruebe ésto como un comando directo en modo **HIRES.**

CURSET 120, 100, 3

entonces

CIRCLE 50, 1

Si Vd. ha determinado anteriormente **PATTERN** a un valor diferente, el círculo será dibujado en puntos, etc.

Pruebe este programa, para un resultado interesante.

```
100  :REM ** CIRCULOS ENLAZADOS **
110  :HIRES
120  :CURSET 120.100,3
130  :FOR N=99 TO 1 STEP-1
140  : CIRCLE N,1
150  :PATTERN 100-N
160  :NEXT N
```

* * * * * * * * *

PUNTOS

Si Vd. quiere saber si un punto en la pantalla está en color de pantalla o de fondo, por ejemplo, para saber si hay un invasor del espacio en el centro de la pantalla, entonces Vd. necesita el comando **POINT**. Para ver como funciona, pruebe lo siguiente en modo directo.

Escriba **HIRES**

Ahora escriba

CURSET 0,0,0 (cursor en la posición **0,0** con color del fondo).
Ahora escriba

TEXT

a continuación

PRINT POINT (0,0)

Como 0,0 está en el color del fondo, obtendrá un 0. Escriba **HIRES.** Esta vez escriba **CURSET 0,0,1** (cursor en la posición 0,0 en color de la pantalla).
Ahora escriba

TEXT,

a continuación

PRINT POINT (0,0).

Esta vez debería obtener -1, ya que el elemento gráfico está en color de primer término.

* * * * * * * * *

FILL (LLENAR)

Es un comando muy útil que puede llenar una área de muchas filas con un valor y muchos caracteres con un solo valor, entre 0 y 127. Hay 200 filas, 40 casillas de caracteres por fila. El valor produce los colores y patrones de acuerdo a los atributos. (vea el apéndice 7 para más detalles).

A continuación verá un programa corto que le muestra las particularidades que el ORIC puede realizar.

```
5     HIRES
10    FOR N= 0 TO 199
20    X= RND(1)*8+16
30    FILL 1,1,X
40    NEXT N
```

La línea 20 escoge los colores de fondo de la pantalla al azar desde la línea 0 (arriba) a la 199 (abajo).

Pruebe de encontrar que otras características puede realizar.

Este programa demuestra las posibilidades de mezcla de colores y otras características.

Observe que la línea 130 tiene un control para evitar la pérdida de sincronismo en la pantalla.

```
100   :REM ** DEMO FILL **
110   :HIRES
120   :REPEAT
130   :A=RND(1)*128+1:IF A . 23 AND A . 32 THEN 130
140   :CURSET RND(1)*90+10,RND(1)*90+10,1
150   :FILL RND(1)*90+1,1,A
160   :UNTIL KEY$ . . ""
```

* * * * * * * * * *

CARACTERES DE DOBLE ALTURA Y DESTELLEANTES

Si Vd. quiere caracteres especiales, o sea, destelleantes o de doble altura, el ORIC tiene una rutina que le permite hacer esto.

Si Vd. mira el apéndice de los atributos del ORIC, verá una tabla que especifica todos los efectos que están disponibles. También necesitará ver la tabla de caracteres de control.

Control D cambia consecutivamente a doble y sencilla altura. . Esto sólo se puede realizar con una sentencia para imprimir. Escriba **PRINT CHR$(4)**

Todo lo que Vd. entre ahora aparecerá dos veces en filas consecutivas. Pruébelo. ¡Socorro!. ¿Cómo lo podemos parar?.

Como es un conmutador alternativo, escribiendo por segunda vez **PRINT CHR$(4)** lo cambiará de nuevo. Los otros caracteres de control se realizan de una manera similar. Pero ... ¡volvamos a los destellos dobles!. El siguiente programa le dará el efecto que Vd. desea.

```
10    PRINT CHR$(12)
20    PRINT CHR$(4); CHR$(27); "N CARACTERES DOBLES"
30    PRINT CHR$(4)
```

No se alarme si parece espantoso; le explicaré línea por línea.

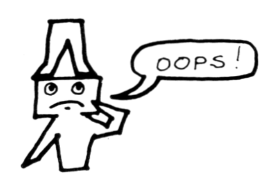
La línea 10 borra la pantalla. También le asegura de que Vd. empieza desde arriba de la pantalla. Verá como esto es importante cuando más adelante añadimos la línea 15.

La línea 20 contiene varias sentencias. **CHR$(4)** conecta automáticamente la doble altura (para ahorrarle escribir dos veces), **CHR$(27)** es el código de escape A.S.C.I.I. (para empezar la secuencia de escape) y la N entre comillas escoge los caracteres de doble altura y destelleantes para el resto de texto -la letra N no se imprimirá.

La línea 30 elimina la doble altura.

Intente cambiar el mensaje entre comillas, alterando el código de salida, o sea, **"J HELLO"** producirá caracteres de doble altura pero no destelleantes.

Cuando haya descubierto como realizar los diferentes efectos, añada esta línea: **15 PRINT** -entonces ejecute el programa. Con este resultado, Vd. comprobará lo importante que es empezar en con un número de línea par (0,2, etc.)



En conclusión, he aquí un programa que usa muchos comandos de gráficos de alta resolución. También muestra como la información de los comandos de gráficos pueden estar en sentencias **DATA.**

```
100   :REM ** PENNY FOR YOUR THOUGHTS **
110   :HIRES
120   :X=100:Y=X
130   :CURSETX,Y,1
140   :PAPER6:INK1
150   :CIRCLE70,1      '**RUEDA DELANTERA**
160   :REPEAT
170   : CURSETX,Y,3
180   : DRAW69*SIN(F),69*COS(F),1
190   : F=F+. 1
200   :UNTIL F> 2*PI
210   :F=0
220   :CURSET 200,140,3
230   :CIRCLE30,1      '**RUEDA TRASERA**
240   :REPEAT
250   : DRAW29*SIN(F),29*COS(F),1
260   : CURSET200,140,3
270   : F=F +. 1
280   :UNTILF> 2*PI
290   :CURSET100,15,3
300   :REPEAT
310   : READ A,B
320   : DRAW A,B,1
330   :UNTIL B = 25
340   :CURSET 160,20,3
350   :FOR N = 1 TO 10 '**TEXT 1**
360   : READ L
370   : CHAR L,0,1
380   : CURMOV 7,8,3
390   :NEXT
400   :CURSET 160,32,3
410   :FOR N = 1 TO 9 '**TEXT2"
420   : READ L
430   : CHAR L,0,1
440   : CURMOV 7,0,3
450   :NEXT
500   :DATA -10,0,10,10,0,20,0,-20,40,0
510   :DATA -10,-10,15,0,-5,10,60,60,0,25
520   :DATA 79,82,73,67,32,82,73,68,69,83
530   :DATA 84,79,32,87,79,82,75,33,33
```

CAPITULO 5
Edición de programas Basic



5. Edición de programas Basic

Cuando Vd. escriba un programa y desee cambiar una línea, existen varios métodos de alterar o borrar instrucciones ya existentes.

Si la línea entera es incorrecta, entonces escribiendo el número de línea seguido de **RETURN** borrará toda la línea.

```
10    PRINT "HOLA"
20    PRINT "EEP!"
30    PRINT "ADIOS"
```

Escriba:-

```
20    RETURN
```

Ahora escriba:-

```
LIST
```

Vd. verá el mismo listado del programa que el anterior, pero esta vez sin la línea 20

```
10    PRINT "HOLA"
30    PRINT "ADIOS"
```

Si Vd. desea borrar todas las líneas en un programa, entonces escriba **NEW**.

Si Vd. está escribiendo en una línea, y descubre que ha cometido un error, Vd. puede borrar el último caracter entrado pulsando la tecla **DEL**. Pulse dos veces la tecla **DEL** y el cursor retrocederá dos posiciones (se mueve hacia la izquierda).

Para borrar una línea completa que Vd. acaba de entrar, mantenga presionada la tecla **CTRL** mientras pulsa la tecla **X**. Aparecerá una barra al final de cada línea, y el cursor irá hasta el principio de una nueva línea.

Si Vd. lista un programa largo, probablemente encontrará que pasa la pantalla demasiado deprisa para leer. Para parar el listado, pulse una sola vez la barra de espacio. Pulsando cualquier otra tecla continuará el listado. Para parar el listado del todo, mantenga presionada la tecla **CTRL** mientras pulsa **C**. Control C también parará la ejecución de todos los programas en Basic. Para continuar escriba **CONT**, a no ser que Vd. haya cambiado algún programa o sus variables. Entonces tendrá que usar **RUN** o **GOTO**.

* * * * * * * * *

COPIAR

Le llevaría mucho tiempo tener que reescribir las líneas enteras, particularmente si contienen información bastante complicada. No le está permitido tener líneas más largas de 78 caracteres. Las líneas de programas muy largas son muy difíciles de leer y estropean el final de los listados. El ORIC le dirá **¡PING!** si trata de exceder este número.

Si Vd. necesita cambiar una línea, el ORIC tiene la facilidad de hacer un **COPY**. Para verlo en acción, escriba este programa corto.

```
10    REM "TEST EDICION"
20    A=20: B=30
30    C= A*B
40    PRINT C
```

Si Vd. decide que la línea 20 debería leerse **A=25: B=5**, y que la línea 30 debería leerse **C=A+B**, esto es lo que Vd. tiene que hacer. Liste el programa y éste aparecerá en la pantalla. Es una buena idea pulsar control L (**CTRL** y **L** a la vez) para borrar la pantalla antes de cada vez que Vd. quiera listar el programa.

Ahora Vd. puede mover el cursor hacia arriba de la pantalla usando las teclas de las flechas situadas al lado de la barra de espacio. El cursor se moverá en la dirección de la flecha. Cuando el cursor esté al lado de la línea 20, mantenga presionada la tecla **CTRL** pulsando a la vez la tecla **A.** El cursor se moverá hacia la derecha y cada caracter por el que pasa será entrado en el buffer de entrada (un almacenamiento temporal).

Cuando el cursor está sobre el 0 del 20, deje de pulsar las teclas **CTRL** y **A**, y pulse **5.** Aparecerá **A5** en lugar de **0.** Continue copiando la línea usando control A hasta que el cursor se coloca sobre el **3** de 30. Deje de pulsar las teclas control A y entre un **5.** El **5** aparecerá en lugar de **3.**

Como Vd. no quiere el **0** en su nueva línea, simplemente pulse **RETURN** y la línea editada quedará almacenada en la memoria del programa. La pantalla mostrará **A=25: B=50,** lo que le puede hacer pensar que Vd. está equivocado. Borre la pantalla como lo ha hecho antes y liste el programa. ¡Voila! -¿Ahora la línea 20 lee **A=25: B=5.**

Como Vd. no copió el **0** al final de la línea, no fue almacenado como parte de la nueva línea. Para cambiar la línea 30, mueve el cursor hasta la línea y entonces copia hasta * usando control A. Entre + entonces copia **B** y pulse **RETURN.**

Recuerde -moviendo el cursor en la pantalla no altera las líneas del programa. Las líneas del programa se entran copiando caracteres ya existentes en la pantalla usando control A, o entrando otros nuevos desde el teclado. Control X le permitirá salir de la línea, las teclas del cursor pueden saltar sobre las letras, **DEL** borrará los errores y **RETURN** entrará la línea nueva.

Hasta que Vd. no esté familiarizado con estas características, borre siempre la pantalla y liste la línea nueva para asegurarse de que ha sido entrada tal como Vd. desea. Descubrirá que Vd. puede editar y copiar líneas muy deprisa y pronto Vd. llegará a ser un profesional usando los diferentes controles de edición del ORIC.

* * * * * * * * * *

TRON y TROFF

Si Vd. está desarrollando un programa en Basic y, a parte de todos sus esfuerzos, no funciona, le da extraños resultados, o simplemente se para la ejecución con un mensaje de error, entonces es útil saber si la capacidad de control del programa es la que Vd. pretendía. El ORIC tiene dos comandos que le permiten hacer esto.

TRON provee de una facilidad de seguimiento que imprime el número de línea que ha sido ejecutado. El propio número de línea se cierra entre corchertes de manera que no pueda producirse un error con la actual pantalla. **TRON** no puede ser entrado directamente por un comando sino que tiene que ser insertado en un programa con el número de línea, o sea:-

```
50   TRON
```

¡A continuación hay un ejemplo de un programa que no funciona!. Escríbalo y ejecútelo.

```
 10   FOR N= 1 TO 4
 20   READ D
 30   ON D GOSUB 100,200,300,400
 40   NEXT N
 50   STOP
100   PRINT "YO"
110   RETURN
200   PRINT "SOY"
300   PRINT "EL ORIC"
310   RETURN
```

```
400     PRINT "QUIEN ES"
410     RETURN
500     DATA 1,2,3
```

Verá que hay algo erróneo con la línea **ON ... GOSUB** cuando se compara con la línea **DATA.**

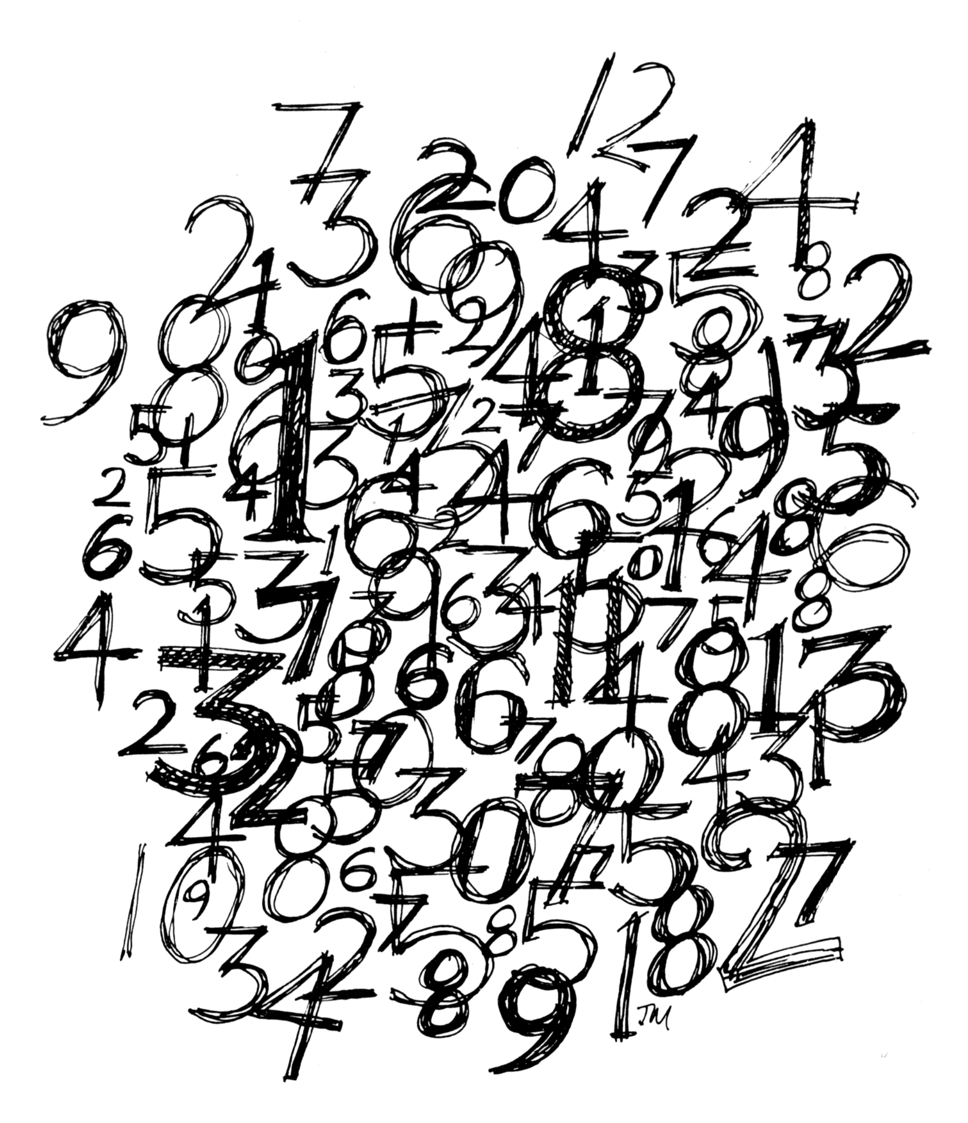
Si Vd. entra

```
5       TRON
```

entonces ejecute el programa, la pantalla se llenará con los números de línea. Vd. puede observar que nunca llegan al 400, y una revisión a la línea **DATA** le revelará por qué no está el 4.

Si Vd. sólo deseaba examinar digamos, el funcionamiento de una subrutina, sería posible comenzar la subrutina con **TRON** (TRAZAR) y terminarla con **TROFF** (NO TRAZAR).

CAPITULO 6
Manipulación de números



6. Manipulación de números

Como ya habrá descubierto, el ORIC puede manejar números muy grandes y también muy pequeños, tanto positivos como negativos. A medida que los números más grandes aumentan, necesitan de más dígitos, por eso 10 necesita dos dígitos, 100 tres, y así sucesivamente. Esto puede resultar complicado y difícil de entender para leer y escribir cuando el número llega ser muy largo. Existe una manera de escribir números que es mucho más compacta, llamada notación científica o exponencial.

10 puede escribirse 1×10^1 (10)
100 puede escribirse 1×10^2 (10×10)
1000 puede escribirse 1×10^3 ($10 \times 10 \times 10$)
y así sucesivamente

El ORIC podría escribir 1×10^3 como 1.00000000E + 3. De hecho, los números hasta 999999999 se ven tal como han sido escritos, ya que el ORIC tiene una precisión de 9 dígitos.

Pruebe estos:-

```
PRINT 999999999 * 1
```

entonces

```
PRINT 9999999999 * 1
```

Esto le muestra como funciona la notación científica, y también como se redondea el número.

Entre números grandes y vea como el ORIC los imprime. También puede entrar números como 2.3E+4 para ver cuales son sus equivalentes. Una manera fácil de recordar como convierten estos números es decir que 2.3E+4 significa
"Escriba 2.3 Mueva las dígitos 4 posiciones a la izquierda. Llene los espacios con ceros".

2.3
23.3 ← 1 posición (ó 10 veces)
230.0 ← 2 posiciones (ó 100 veces)
2300.0 ← 3 posiciones (ó 1000 veces)
23000.0 ← 4 posiciones (ó 10000 veces)

Así pues 2.3E+4 es lo mismo que 23000

2.3E-4 ¿Qué significa? El signo negativo después de E no significa que el número es negativo, simplemente muy pequeño. Significa:-"Escriba 2.3 Mueva los dígitos 4 posiciones a la derecha. Llene los espacios con ceros.

2.3	
0.23	1 posición (o dividido por 10)
0.023	2 posiciones (o dividido por 100)
0.0023	3 posiciones (o dividido por 1000)
0.00023	4 posiciones (o dividido por 10000)

Así pues, 2.3E-4 es un número muy pequeño: 0.00023. Si el número es un número negativo pequeño, debería escribirse -2.3E-4 Asegúrese de entender estas diferencias si es que desea comprender como el ORIC maneja los números.

* * * * * * * * *

INT

INT es una función que retorna el número entero más grande menor que o igual al valor entre paréntesis. Pruebe estos para ver si el ORIC contesta las respuestas adecuadas:-

PRINT INT (1.5)
PRINT INT (2)
PRINT INT (-2)
PRINT INT (-1.5)

Observe particularmente el resultado del último ejemplo **-INT** siempre redondea un número menor que el que está entre paréntesis, a no ser que ya sea un entero.

* * * * * * * * *

ABS

ABS retorna el valor absoluto de un número. Si es positivo, permanece como tal. Si es negativo, se convierte en positivo.
Pruebe estos:-

PRINT ABS (4.3)
PRINT ABS (-4.3)

* * * * * * * * *

SGN

SGN retorna -1, 0 ó 1, según si el valor que está entre corchetes es negativo, un cero, o positivo. Pruebe estos para ver como funciona:-

```
10    FOR N= -5 TO 5
20    PRINT N, SGN(N)
30    NEXT N
```

* * * * * * * * *

DATA

Si Vd. tiene que utilizar muchos números en un programa entonces es posible almacenarlos en el programa como **DATA**, mucho mejor que tenerlos que escribir cada vez. Este pequeño ejemplo muestra como incorporar esta información en sus programas:-

```
10    FOR N=1 TO 5
20    READ A
30    S=S+A
40    NEXT N
50    PRINT"SUM="S
60    DATA 1,3,8,6,4
```

La línea 20 lee los datos, un elemento cada vez, y asigna ese valor a la variable **A.** Cada vez se le añade una S (inicialmente cero) y se imprime en la línea 50.

Cuando se ejecuta el programa, un puntero se mueve a lo largo de cada elemento mientras lee, y permanece en el último elemento que alcanza. Si Vd. escribe **GOTO 10,** el puntero no se recupera, y obtendrá un mensaje de error **OUT OF DATA. RESTORE** es un comando que recupera el puntero en un programa. Añada **15 RESTORE** y vea el efecto que tiene en el programa. El puntero se recupera al primer elemento **DATA** cada vez que se ejecuta el bucle, de manera que cada vez se añade una S, y se ignora el resto de datos.

* * * * * * * * *

ARRAYS (MATRICES)

Algunas veces es mejor juntar variables similares, en lugar de darles nombres diferentes. Las matrices llevan paréntesis a continuación del nombre de la variable de manera que se pueda identificar un elemento. Ejemplo: **N(1), N(2), N(3),** y **N(4)** son todos los elemntos de la matriz **N.**

El ORIC guarda automáticamente espacio de hasta 10 elementos en una matriz. Si Vd. necesita más espacio, tendrá que usar la sentencia **DIM,** ejemplo: **DIM N(14)** guardará espacio para 15 elementos. (Las matrices empiezan con cero, no con uno, como en la mayoría de las formas del Basic).

La razón de usar las matrices en lugar de variables simples es que se pueden meter en bucles **FOR/NEXT** más fácilmente.

```
10   FOR N=1 TO 5
20   A(N)=N*N
30   NEXT N
40   FOR X=1TO 5
50   PRINT X, A(X)
60   NEXT X
```

Las líneas 10 a la 30 cargan la matriz **A(N)** con los cuadrados del número del bucle **(N)**. Las líneas 40 a la 60 imprimen los contenidos de la matriz en la pantalla. Observe que no es necesario llenar cada elemento en una matriz. Los elementos "vacíos" contendrán cero.

Una matriz como las que hemos visto es similar a una columna de números, pero también es posible tener filas y columnas -en otras palabras, una matriz bidimensional. Las matrices dimensionales deben ser dimensionadas antes de ser utilizadas.

```
10   DIM A(5,5)
20   FOR N=1 TO 5
30   FOR M=1 TO 5
40   A(N,M)=N*M
50   NEXT M,N
```

Esto cargará números en 25 posiciones. Se puede apreciar que este programa genera una tabla, contenida en la matriz, en la que se hagan los resultados de algunas multiplicaciones simples.

N

M \ N	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	1	2	3	4	5
2	0	2	4	6	8	10
3	0	3	6	9	12	15
4	0	4	8	12	16	20
5	0	5	10	15	20	25

Contenidos de una matriz a(N,M) después de la ejecución de un programa.

Los números entre corchetes identifican el elemento al que Vd. desea referirse, y se les conoce como subíndices. Ejemplo: **A(2,3)** es 6 y **A(5,5)** es 25.

Si Vd. desea tener más de una dimensión, es perfectamente posible. Sin embargo, es importante recordar que **DIM N(10,10,10)** tiene 1000 elementos, ¡y Vd. puede quedarse muy rápido sin memoria suficiente!

* * * * * * * * * *

LOGS

He mencionado que los números pueden escribirse usando notación científica. Ej.: 1.6E+2 es lo mismo que 160. Más normal es que 1.6E+2 se escriba como $1.6x10^2$ que significa que 1.6x10x10 ó 1.6x100. El 2 pequeño en 10^2 significa "escriba el número más grande tantas veces como indique el número pequeño multiplicándolos juntos". O sea, 10x10.

Así pues, 10^4 significa 10x10x10x10 ó 1000. El número más pequeño se llama índice o exponente. 10^4 se lee normalmente "10 elevado a la potencia de 4" o sólo "10 a la potencia de 4".

Esto nos introduce a los logaritmos ,o más corto logs, porque el log de un número es la potencia a la cual 10 debe ser elevado para producir ese número, Ej.: 4 es el log de 10.000.

Para encontrar el log de un número entre el 1 y el 10, tenemos que encontrar el índice que dará ese número. Debe ser entre 0+1. Si Vd. escribe

PRINT LOG (5)

entonces dará el resultado del logaritmo. Vd. debería de obtener

0.6989700004

Para revisar si es realmente el logaritmo de 5, intente elevar 10 a esa potencia y vea si da como resultado 5.

Escriba

PRINT 10 0.698970004.

y verá si Vd. estaba en lo cierto. (El signo ↑ significa "a la potencia de" y es SHIFT 6 del teclado). Recuerde $10^{\log x} = X$.

Al igual que logaritmos comunes o de base 10, también hay en el ORIC logaritmos naturales (neperianos). Escriba

PRINT LN(5)

Vd. debería obtener

1.60943791

que es el logaritmo natural de 5. ¿Cómo entrar más de un logaritmo? Los logaritmos de base 10 son la potencia a la cual 10 debe obtener ese número. Los logaritmos naturales son la potencia a la cual e debe ser elevada para obtener ese número. e es

2.718281828

e es el resultado de estas series

$$e = 1 + 1 + \frac{1}{2} + \frac{1}{2x3} + \frac{1}{2x3x4} \ldots \text{etc.}$$

El exponente natural de un número es el inverso del log de un número natural y puede obtenerse escribiendo PRINT EXP (X)
Así pues X = EXP (LN(X))





Para obtener el logaritmo de un número a otra base, use la fórmula

LOG base z (X) = LOG_e (X)/LOG_e (Z)

Naturalmente el logaritmo e es lo mismo que LN en el ORIC.

* * * * * * * * * *

BASES DE LOS NUMEROS

Hasta aquí hemos encontrado números binarios y decimales. Quizás Vd. está confuso de tantas maneras diferentes de presentar las mismas cantidades. De hecho, no son tan difíciles de manejar como parecen -todo depende de como escoger los grupos de números. Generalmente, los agrupamos de diez en diez -

probablemente porque tenemos diez dedos -no hay ninguna otra razón. Nuestro sistema completo de contar usa grupos de diez. Cuando tenemos diez grupos de diez, forma un centenar, y diez centenares forma un millar y así sucesivamente.

Ej.: 3742 está formado de 3 miles, 7 centenas, 4 decenas y 2 unidades. La figura más larga que Vd. puede tener en cualquier columna es 9. Una más, y ya tiene suficiente para formar un grupo de 10 en la siguiente columna.

Ej.:
```
  9
 +1
 --
 10
 --
```

Imagine que los humanos tuvieran 8 dedos. Hubieran utilizado estos símbolos: 0,1,2,3,4,5,6,7,8 y el 9 no existiría.

Ej.:
```
  7
 +1
 --
 10
 --
```

El dígito más grande que Vd. ahora puede tener en una columna es 7. Una más forma un grupo de ocho, de manera que la respuesta es 10. Esto no se lee como "diez" sino "uno-cero" en "base ocho". Para mostrar que no es un número normal (o decimal o denario o base 10), entonces es normal escribirlo 10_8.

Al igual que cabeceras de columna en base 10, los números se representan en decenas,

Ej.:	miles	centenas	decenas	unidades
	(10x10x10)	(10x10)	(10)	

así pues en base ocho, las cabeceras se representan en ochos,

Ej.:	512	64	8	unidades
	(8x8x8)	(8x8)	(8)	

de manera que 1241_8 es lo mismo que
(1x512) + (2x64) + (4x8) + 1 = 673 en base 10.

En el lenguaje de la computadora, binario o base 2, aplican las mismas reglas -pero ahora hay sólo 2 dígitos, 0+1 y los grupos son en dos.

Ej.:	16	8	4	2	unidades
	(2x2x2x2)	(2x2x2)	(2x2)	(2)	

Así pues en binario, el número 10111 es lo mismo que

(1 x 16) + (0 x 8) + (1 x 4) + (1 x 2) + 1 = 23 en base 10.

Esto puede producir algo de confusión cuando hay números largos. De hecho, Vd. necesita 8 dígitos para obtener 255, y 65535 es 1111111111111111 -¡dieciséis dígitos!

Cabría la posibilidad de ignorar el código binario para números largos y saltar a base 10, pero esto no da ninguna pista de como se almacena el número en una computadora.

Se llega a un compromiso usando la base dieciséis (o hexadecimal o hex que es como se llama normalmente). La base 16 necesita 16 dígitos, de manera que las letras se usan después del 9. Esto significa que contando en hexadecimal da como resultado esto:-

0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

Vd. puede ver que quince en hexadecimal es F, y dieciséis es 10. Esto da las siguientes cabeceras de columnas:-

4096	256	16	unidades
(16x16x16)	(16x16)	(16)	

Así pues 12AF en hexadecimal, es lo mismo que
(1 x 4096) + (2 x 256) + (10 x 16) + 15 = 4783 en base 10.

¿Por qué escogemos un sistema de numeración tan horroroso? Quizaś lo ha adivinado -a primera vista muestra como los números son almacenados en una computadora, cogiendo cada vez bloques de 4 bits.
F0 puede ser almacenado en un byte

$$\underbrace{1111}_{F}\quad\underbrace{0000}_{0}$$

El número binario que es el equivalente de 65535 en base 10 es FFFF en hexadecimal. Ahora Vd. puede ver por que 65535 es la posición de la memoria más alta que Vd. puede direccionar usando dos bytes.

El ORIC reconocerá los números hexadecimales como tales, precedidos por una #, llamada "almohadilla". Así pues **PRINT# 1A** forma el número 26. Intente para Vd. mismo algunas conversiones. El número más largo que Vd. puede convertir es **# FFFF**. Revise sus resultados con la tabla en los apéndices.

NOTA: Vd. puede encontrar algunos libros que identifican los números hexadecimales usando $, pero el ORIC interpretará esto como un string (variable alfabética).

Si Vd. necesita convertir números hexadecimales a base 10, usará #. Ej.:

PRINT#10

obtendrá 16.

Para convertir una string en hexadecimal o base diez conteniendo el valor y precedido por una #, escriba

PRINT HEX$(16)

Esto producirá #10. Lo mismo que con #, hay un límite máximo e 65535. Ej.: **# FFFF**. Este programa imprimirá en base 10 números hasta el 255 y también su equivalente hexadecimal.

```
10    FOR N= 0 TO 255
20    PRINT N, HEX$(N)
30    NEXT N
```

* * * * * * * * * *

RUTINAS MATEMATICAS

Aunque no conviene pensar de las computadoras como puramente manipuladoras de números, no cabe duda de que hacen muchas operaciones matemáticas que normalmente son aburridas y de repetición, comparativamente simples. El Oric las hahecho en rutinas que se llaman ""empaquetado de números".

Si Vd. necesitara conocer todas las raíces cuadradas desde 1 a 100, Vd. perderá mucho tiempo mirando las tablas o pulsando teclas de una calculadora. El Oric puede hacer ésto mucho más simple. Escriba este programa:-

```
10   CLS
20   FOR N=1 TO 100
30   PRINT N, SQR(N)
40   NEXT N
```

Si Vd. lo ejecuta, los números del 1 al 100 destellearán en el borde izquierdo de la pantalla, con sus raíces cuadradas cerca de ellos en el centro de la pantalla. El Oric las calcula tan rápido que Vd. lo encuentra difícil para leer. Ponga

35 WAIT 10

para parar el programa.

El Oric puede calcular raíces sin usar el SQR. Hay un método para encontrar raíces llamado método iterativo de Newton-Raphson. Iteración significa repetir una operación una y otra vez - un uso ideal para un bucle en el Oric. Cada vez, la respuesta se va aproximando hasta conseguir una más correcta. Este programa corto muestra las aproximaciones, y se para cuando la respuesta es correcta. (La línea 80 salta del bucle si la respuesta está en .0.000001 de la respuesta verdadera -sólo en el caso de que no haya una respuesta correcta).

```
 5   REM *** RAIZ ITERATIVA ***
10   IMPUT "DIME UN NUMERO";S
20   INPUT "SACAR LA RAIZ";G
30   PRINT G
40   X=S/G
50   G=(X+G)/2
60   R=G*G
70   IF R< (S+0.000001) AND R >(S-0.000001) THEN GOTO
     90
80   GOTO 30
90   PRINT "RAIZ=";G
```

Existe otro uso para el cerebro matemático de alta velocidad del ORIC. El matemático Leibnitz del siglo 17, quien hizo

posible los cálculos, descubrieron una manera para calcular .. π
π es un número irracional; en otras palabras, nunca puede calcularse a número finito de decimales.

Leibnitz descubrió que esta secuencia se acercaba más al valor exacto cada vez

$$\pi = 4\left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} \cdots\right)$$

Vd. seguramente verá una secuencia regular en estas fracciones. El ORIC ama las secuencias regulares, ya que pueden ponerse en los bucles. Si Vd. trataba de calcular la fórmula de arriba con lápiz y papel hubiera tardado mucho tiempo- hasta - 1/9. - mucho peor cuando Vd. se da cuenta de que la respuesta no está lo suficientemente clara hasta después de haber hecho bucles varios centenares de veces. Pobre Leibnitz, ¡pero que suerte para Vd! Pruebe este programa.

```
 5   REM *** PI LENTO ***
10   CLS
20   DEF FNA(N)=(-1/(N+2))
30   FOR X=3 TO 10003 STEP 4
40   S=S+FNA(X)
50   APROX= 4*(1+S)
60   PRINT APROX
70   NEXT X
```

La línea 10 define una función, **A**, que contiene una variable **N.** Esto le ahorra de tener un montón de líneas más adelante. **FNA** calcula las series y se llama en la línea 30. El bucle sube en apartados de 4, empezando con 3, de manera que X es 3,7,11,15 etc. Esto realiza incrementos correctos y el resultado se imprime en la línea 50. Si Vd. ejecuta el programa, el Oric imprime un resultado paraπ que se acerca más y más al resultado correcto. Compárelo con 3.1416 y verá que no es la manera más rápida para calcularπ, ni tampoco con el ORIC.

Para encontrar . más deprisa, simplemente escriba

PRINT PI

Esto le dará un valor más seguro de n para diferentes posiciones de decimales, ya que **PI** queda almacenado como una constante por el ORIC.

Recuerde que esto significa que Vd. no puede escoger **PI** como nombre de una variable, o cualquier palabra empezando con **PI,** como **PIG=8** o **PIPES=78,** porque **PI** es una palabra reservada.

* * * * * * * * * *

NUMEROS ALEATORIOS

Hay una función útil en el ORIC que a menudo se usa en programas de juegos. Es la **RND**, la cual retorna un número seudoaleatorio. Debido a la forma en que las computadoras generan números al azar, esto no será verdaderamente aleatorio, y sería posible descubrir un patrón de series de números producidos. Esto no parece ser muy obvio a menos que Vd. realice un análisis estadístico en las series, de manera que **RND** para todos los propósitos normales está considerado totalmente al azar.

Si Vd. no está seguro de lo que son los números al azar, entonces considere un dado. Tiene una oportunidad igual de producir números del 1 al 6. El orden en que los números están realmente producidos en una serie de tiradas es al azar. Para simular esto en el ORIC, pruebe este programa.

```
5    REM *** TIRADOR DE DADOS ***
10   FOR N= 1 TO 10
20   PRINT "PULSA UNA TECLA PARA TIRAR EL DADO"
30   GET A$
40   A= INT(RND(1)*6)+1
50   PRINT A
60   NEXT
```

La línea 20 espera que se pulse cualquier tecla. La línea 30 escoge un número al azar entre 0+1, multiplicado por 6, la función **INT** pierde cualquier fracción decimal, y finalmente, se añade un 1. Esto hace que se produzca un número del 1 al 6.

RND(n) producirá un número al azar mayor que o igual a 0 y menor que 1, si **n** es un número positivo. Si **n** es un número negativo, entonces la secuencia al azar se fija a un número particular, y las subsiguientes **n** positivas siempre producirán la misma secuencia. Si **n** es cero, el último número generado al azar será producido.

* * * * * * * * *

Para concluir este capítulo, hay un programa que utiliza muchas funciones mencionadas en este capítulo. También usa alguna string conteniendo rutinas que Vd. no puede entender del todo hasta que no haya leído el siguiente capítulo. Si es necesario vuelva a él más adelante.

Los calendarios son difíciles de construir debido a las posiciones de la tierra al no tomar un número exacto de días para dar la vuelta al sol. De hecho, tarda 365.242216 días para hacer un año exacto.

Varias personas, desde Numa Pomilius y Julio César hasta el Papa Gregorio, han intentado corregir el calendario, pero todavía no es perfecto.

Todo esto hace más complicado calcular en que día de la semana cae una fecha en particular. El matemático alemán, Gauss, realizó una fórmula que funciona para cualquier fecha desde 1752, cuando el calendario Gregoriano fue iniciado en el Reino Unido y colonias Americanas.

```
5    REM *** CALCULADOR DEL DIA ***
10   CLS
20   PRINT "ENTRA EL DIA, MES Y AÑO"
30   INPUT "DIA";D
40   IF D <1 OR D> 31 THEN 30
50   INPUT "MES";M
60   IF M <1 OR M> 12 THEN 50
70   INPUT "AÑO";Y
80   IF Y <1752 OR Y> 8000 THEN 70
90   M=M-2: IF M <1 THEN M=M+12:Y=Y-1
100  Y$=STR$(Y)
110  C=INT(Y/100)
120  Y= VAL(RIGHT$(Y$,2))
130  A=INT(2.6*M-0.19)+D+Y+INT(Y/4)+INT(C/4)-C*2
140  DIA= INT((A/7-INT(A/7))*7+0.1)
150  DIA=DIA+1
160  FOR N=1 TO dia
170  READ DIA$
180  NEXT N
190  PRINT DIA$
200  DATA DOMINGO, LUNES, MARTES, MIERCOLES,
     JUEVES, VIERNES, SABADO
```

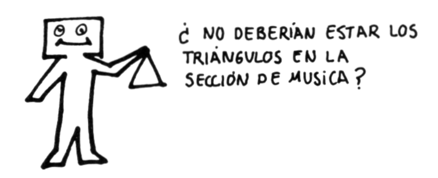
CAPITULO 7
Más funciones matemáticas



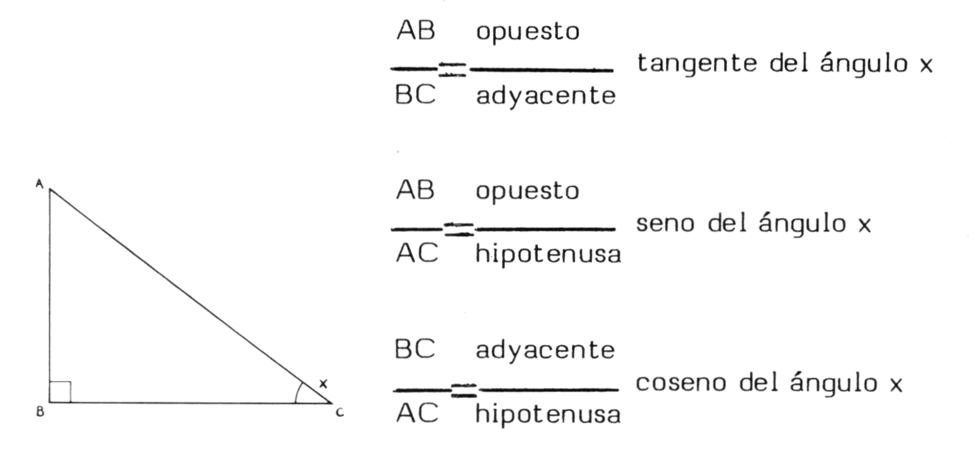
7. Más funciones matemáticas

TRIGONOMETRIA

El ORIC tiene muchas funciones que Vd. reconocerá si tiene una calculadora científica o si Vd. recuerda sus lecciones de geometría en la escuela. Son SIN(senos) COS(cosenos), TAN(tangentes). Son cocientes de longitudes de lados de triángulos para ángulos diferentes.

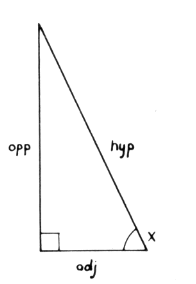


ABC es el ángulo derecho del triángulo, x es el ángulo de ABC. El lado AB es el opuesto al ángulo x, BC es el adyacente al ángulo x, y AC se llama hipotenusa.



Vd. probablemente puede ver como la cantidad cambia en estos tres ejemplos de triángulos:-

1.) Cuando x es grande

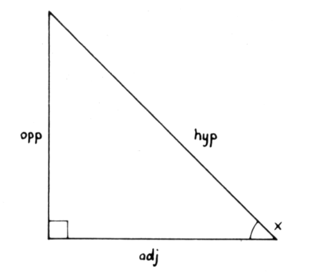


$TAN(x)=\frac{opu}{con}$ → un número grande que va al al infinito ya que x es casi de 90º

$SIN(x)=\frac{opu}{hip}$ → va hacia 1 ya que x es casi de 90º

$COS(x)=\frac{con}{hip}$ → va haci 0 ya que x es casi de 90º

2.) Cuando x es 45º

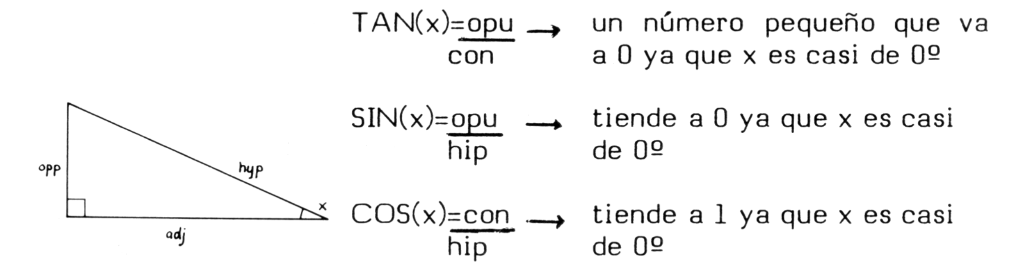


$TAN(x)=\frac{opu}{con}$ → 1

$SIN(x)=\frac{opu}{hip}$ → $\frac{1}{\sqrt{2}}$

$COS(x)=\frac{con}{hip}$ → $\frac{1}{\sqrt{2}}$

3.) Cuando x es pequeña



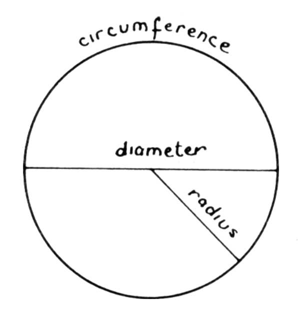
$TAN(x)=\frac{opu}{con}$ → un número pequeño que va a 0 ya que x es casi de 0º

$SIN(x)=\frac{opu}{hip}$ → tiende a 0 ya que x es casi de 0º

$COS(x)=\frac{con}{hip}$ → tiende a 1 ya que x es casi de 0º

Vd. puede obtener TAN, SIN &COS escribiendo simplemente **PRINT TAN (x),** etc. El único problema es que el Oric, como la mayoría de las computadoras, le gusta los ángulos en radianes, no grados. Los grados pueden cambiarse muy fácilmente en radianes y viceversa.

Un breve resumen sobre los círculos.



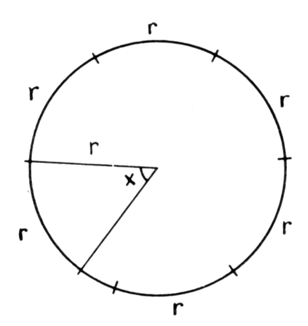
diámetro = radio x 2

$$\pi = \frac{\text{circunferencia}}{\text{diámetro}}$$

$$\pi = \frac{\text{circunferencia}}{\text{radio} \times 2}$$

$\therefore$ circunferencia=radio x π x 2
y radio = $\frac{\text{circunferencia}}{2 \times \pi}$

¿Cuántas veces el radio debería caber alrededor de la circunferencia? Más de 6 veces -de hecho 2x. veces. Si Vd. tuviera que cortar un tajo del círculo, de manera que la parte curvada sea igual al radio, r, entonces el ángulo en el centro x, es 1 radian. Un cículo completo = 360º de manera que 1 radian = 360º = $\frac{57.29578º}{2\pi}$



Para convertir radianes a grados, use la fórmula:-

$$\text{grados} = \frac{\text{rads} \times 360}{2\pi} \simeq \text{rads} \times 57.29578$$

Para convertir grados a radianes, use la fórmula:-

$$\text{rads} = \frac{2 \times \pi \times \text{grados}}{360º} \simeq \frac{\text{grados}}{57.29578}$$

A continuación verá algunos programas para mostrar como el ORIC puede usar funciones trigonométricas para dibujar en la pantalla, lo mismo que calcular para Vd.

PROG. 1. Ondas del seno

```
  5   REM *** SENO ***
 10   HIRES
 20   DRAW 0,199,1
 30   CURSET 0,100,3:DRAW 239,0,1
 40   FOR A= PI TO STEP 0.02
 50   CURSET A*38+120,SIN(A)*99,1
 60   NEXT
 70   PRINT "CURVA SENOIDAL"
 80   GET A$
```

Esto dibuja una curva del seno desde $-\pi$ a π La línea 70 imprime en la línea 3 texto en la pantalla y lo mantiene hasta que se pulsa cualquier teclas en la línea 80. Cambie el SIN en la línea 50 por COS y vea la diferencia mientras el ORIC dibuja la curva de un COSENO.

PROG. 2. Torre

```
  5   REM *** TORRE ***
 10   HIRES
 20   CURSET 20,20,3
 30   DRAW 0,160,1
 40   DRAW 200,0,1
 50   DRAW-200,-160,1
 60   CURSET 25,170,3
 70   A$="↑ TORRE"
 80   FOR N=1 TO 6
 90   CHAR ASC(MID$(A$,N,1)),0,1
100   CURMOV 8,0,3
110   NEXT
120   CURSET 200,170,3
130   CHAR ASC("X"),0,1
140   INPUT "DISTANCIA";D
150   INPUT "ANGULO X(GRADOS)";X
160   XR = X/57.29578
170   H= TAN(XR)*D
180   PRINT H;
```

Esto calcula la altura de una torre, si Vd. puede dar la distancia y el ángulo desde su posición hasta el final de la torre.

La línea 30 a la 50 dibuja el triángulo, las líneas 70 a la 110 son necesarias para imprimir en la pantalla de alta resolución, y la línea 160 convierte los grados a radianes. Observe el punto y coma en la línea 180. Esto guarda la respuesta a la vista en la pantalla.

Hay una lista de funciones derivadas en los Apéndices. Vd. las puede definir usando **DEF FN,** o por el uso del caracter definido **&** (vea el capítulo Código de Máquina) como una función de extensión.

CAPITULO 8
Palabras



8. Palabras

Más adelante, descubrimos que las computadoras pueden manipular cualquier colección de símbolos, no sólo números. Así pues el ORIC sabe que los símbolos deben ser considerados como tales, y no los confunde por variables, tienen que estar encerrados entre comillas.

Así pues,

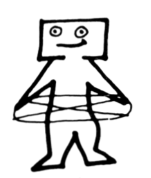
PRINT **A**

imprimirá un cero, ya que es considerado como una variable.

PRINT **"A"**

imprimirá una **A**, ya que está encerrada entre comillas.

Las variables de cadenas alfanuméricas son identificadas por el signo del dólar al final. Ej.: **A$** ó **A3$**. Recuerde que el ORIC sólo lee los primeros dos caracteres de un nombre de una variable, de manera que **BIG$** es lo mismo que **BIKE$**.



Las cadenas son asignadas usando **LET**, aunque es opcional.

LET **A$="HOLA"**

es lo mismo que

A$="HOLA"

El único operador matemático simple que puede ser utilizado con las cadenas alfanuméricas es +. Así pues,

```
10   A$="HOLA"
20   B$=A$+A$
30   PRINT B$
```

imprimirá **HOLAHOLA**.

La línea 20 no podría escribirse como **B$ = 2*A$.** La longitud total de una cadena no debe exceder los 255 caracteres.

Para encontrar la longitud de una cadena alfanumérica hay una función llamada **LEN.**

IF A$="HOLA" THEN PRINT **LEN(A$)**
imprimirá **4.** El valor puede ser asignado a una variable.

```
10    INPUT A$
20    L=LEN(A$)
30    PRINT A$;"CONTIENE";L;"CARACTERES"
```

Aunque una cadena no puede utilizarse como un número directamente, es posible convertirla en un número, usando la función **VAL.**

```
10    A$="56"
20    V=VAL(A$)
30    PRINT V
```

Como **V** no es una variable de una cadena, puede ser tratada como un número, y **PRINT2*V** dará un valor de 112. Si el primer caracter en la cadena es un caracter alfabético entonces da un valor de cero.

```
10    A$="ORIC"
20    V=VAL(A$)
30    PRINT V
```

Hay una función que funciona en dirección opuesta. **STR$** convierte una expresión numérica en una cadena alfanumérica.

```
10    A=128
20    A$=STR$(A)
30    PRINT A$
```

Vd. no puede decir la diferencia entre **PRINT A** y **PRINT A$** - aparecen los mismos resultados. Sin embargo, **PRINTA+A** dará **256,** cuando **PRINT A$+A$** dará **128 128** porque el ORIC los trata diferente.

Si Vd. mira al final de este libro, encontrará una tabla titulada códigos ASCII. También se mencionan en el Capítulo 4.

Usando la función **ASC** le dará el código para cualquier caracter del teclado. La función **CHR$** funciona en dirección opuesta, y convierte un número entre 32 y 128 en el caracter correspondiente. Para listarlos a todos, ejecute este programa.

```
10   FOR N=32 TO 128
20   PRINT "EL CODIGO ASCII";N;"PERTENECE
     A";CHR$(N)
30   WAIT 20
40   NEXT N
```

Como todos los caracteres tienen códigos **ASCII**, pueden ser almacenados en orden. Como Vd. puede ver, el orden numérico es lo mismo que el alfabético, de manera que Z, que tiene el valor **90**, es mayor que **A** que tiene el valor **65.**

Vd. puede usar los signos mayor que (>) y menor que (<) para comparar cadenas, igual que números. Sin embargo, debe de tener cuidado de evitar de mezclar mayúsculas con minúsculas, y todas las letras en minúsculas tienen valores mayores que las mayúsculas, así pues **"manzana"** es menor que **"cebra"**, **"manzana"** es mayor que **"Cebra".**

Para que Vd. aprenda a manejar las cadenas, existen tres funciones más muy útiles - RIGHT$, **LEFT$** y **MID$. RIGHT$** retorna la parte derecha de una cadena de la siguiente manera:

```
A$="ABCDEFGHIJ"

PRINT RIGHT$(A$,2)
```

imprimirá **"IJ".** El número 2 es la cantidad de caracteres que salen como resultado.

LEFT$ da la parte izquierda de una cadena de la siguiente manera:

```
A$=ABCDEFGHIJ"

PRINT LEFT$(A$,2)
```

imprimirá **"AB"**

MID$ necesita un poco más de información.

```
A$= "ABCDEFGHIJ"
PRINT MID$(A$,5,2)
```

imprimirá **"EF".** El ejemplo significa -retorna los 2 cracteres de la cadena **A$**, empezando en la posición 5. El segundo número puede

ser omitido, en este caso todos los caracteres a la derecha de, e incluidos, el primer número son retornados.

```
A$="ABCDEFGHIJ"
PRINT MID$(A$,5)
```

imprimirá **"EFGHIJ".**

A continuación verá un programa corto que muestra estas funciones.

```
10   PRINT "ENTRAR UN LITERAL"
20   INPUT A$
30   IF LEN (A$) . 3 THEN PRINT "DEMASIADO
     CORTO":GOTO10
40   PRINT A$ "TIENE";LEN(A$);"CARACTERES"
50   PRINT "EMPIEZA CON";LEFT$(A$,1)
60   PRINT "Y TERMINA CON";RIGHT$(A$,1)
70   PRINT "Y TIENE";MID$(A$,2,LEN(A$)-2);"EN MEDIO"
```

A menudo es más fácil tener información almacenada en un programa. Lo mismo que es posible contener números en sentencias **DATA**, algo parecido les pasa a las cadenas alfanuméricas.

```
10   FOR X=0 TO 3
20   READ NOMBRE$(X)
30   PRINT NOMBRE(X)
40   NEXT
50   DATA TONI, TERESA, DANI, MARTIN
```

Los datos se leen una sola vez por el nombre y son almacenados en una matriz, **NAME$. NAME$(0)** es entonces **TOM, NAME$(1)** es **TERESA, NAME$(2)** es **DENIS,** y **NAME$(3)** es **MARTIN.**

Las matrices de cadenas alfanuméricas son parecidas a las matrices numéricas. El ORIC reserva automáticamente espacio de hasta 11 elementos (numerados del 1 al 10). Si Vd. quiere más, entonces debe poner el número que Vd. necesita en una sentencia **DIM.** Ej.: **DIM A$ (19)** crearía espacio para 20 elementos.

Hay una cosa especial a observar sobre las cadenas en las líneas de **DATA.** Si Vd. inserta espacios en blanco, serán ignorados por el ORIC. Ej.:

DATA AB, C,**DE**

perderá el espacio antes de la **C.** Si Vd. quiere incluir el espacio, debe encerrar la letra entre comillas. Ej.:

DATA AB, " C", DE

Cuando Vd. ejecuta un programa, el puntero de **DATA** se va hasta el primer elemento de la línea **DATA** y lee desde allí. Si Vd. ejecuta el programa de los nombres, el puntero del programa leerá los 4 elementos, entonces se queda al final. Entonces si Vd. entra

GOTO **10**

el puntero no puede encontrar más elementos, por eso aparece el mensaje de error

OUT OF DATA IN **20**

Para volver a enviar el puntero en el programa, use el comando **RESTORE**. Si Vd. añade esta línea al programa, encontrará que puede usar **GOTO 10** sin obtener ningún mensaje de error.

35 RESTORE

Como está en el bucle, cada vez se ignorará el puntero, y **TOM** quedará grabado en todos los elementos de la matriz.

* * * * * * * *

CLASIFICACION

Para terminar este capítulo, a continuación hay un programa que muestra varias técnicas del ORIC para el manejo de strings, particularmente el uso de las matrices de string y la comparación de strings.

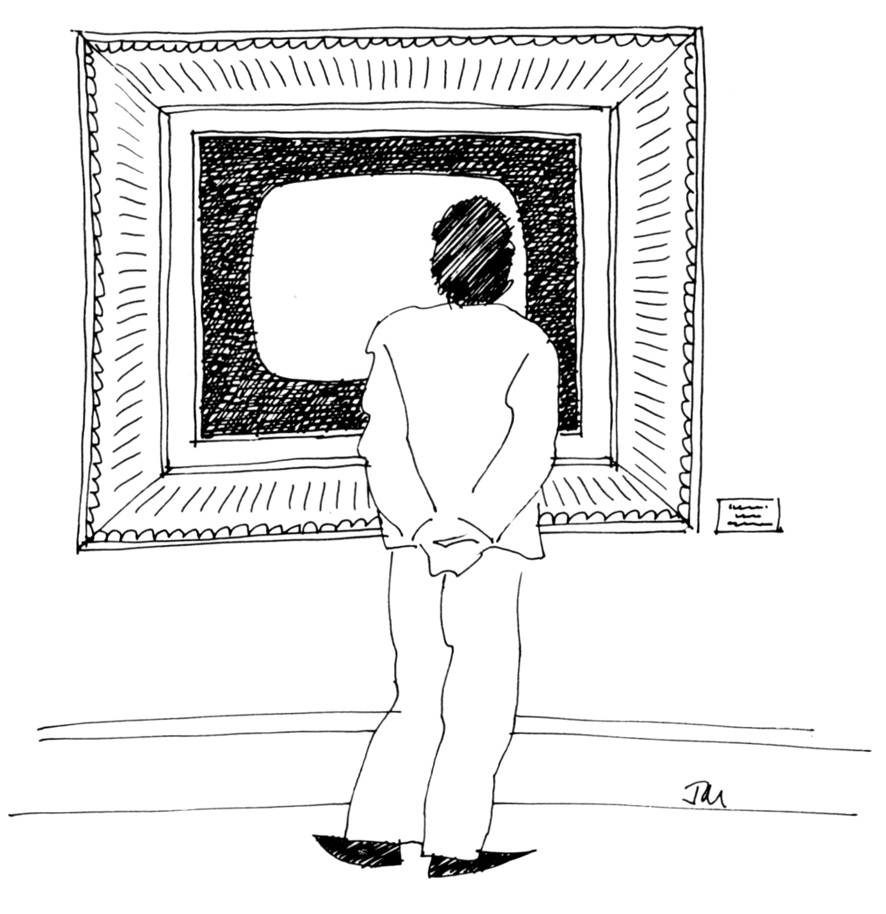
Hay muchas clases de métodos de clasificación que pueden usar las computadoras. Todos funcionan basándose en que las palabras que empiezan con letras que tienen códigos bajos en **ASCII** terminan la rutina al principio de una matriz, las que tienen códigos altos en **ASCII** al final, y el resto clasificadas en orden numérico y por lo tanto alfabético.

Este programa graba las palabras que se han de clasificar dentro de una matriz **A$**, y usa **U$** como un almacén temporal mientras se está listando. **A$** se va barajando hasta que las palabras son ordenadas correctamente, cuando se imprimen las líneas de la 150 a la 170.

Vd. podría usar este programa como una subrutina de clasificación en sus propios programas y renumerarlos desde, digamos, 2000.

```
  5    REM ***CLASIFICACION***
 10    INPUT "NRO. DE PALABRAS";N
 15    DIM A$(N+1)
 20    FOR X= 0 TO N-1
 30    INPUT A$(X)
 40    NEXT
 50    FOR X= 0 TO N-1
 60    PRINT A$(X):NEXT
 70    FOR K=0 TO N-1
 80    FOR L=K+1 TO N
 90    IF A$(L)> =A$(K) THEN 130
100    U$=A$(L)
110    A$(L)=A$(K)
120    A$(K)=V$
130    NEXT L
140    NEXT K
150    FOR X=0 TO n
160    PRINT A$(X)
170    NEXT
```

CAPITULO 9
Gráficos avanzados



9. Gráficos avanzados

GRAFICOS DEFINIDOS POR EL USUARIO

Cuando se conecta el ORIC, tanto el conjunto estandard de caracteres como el alternativo son grabados en memoria RAM. El conjunto de caracteres estandard contiene todos los caracteres normales del ASCII tal como se ve en el apéndice y el conjunto alternativo de caracteres contiene gráficos de teletext. Ambos conjuntos pueden ser completa o parcialmente reescritos.

En un juego, Vd. puede querer usar texto, y también unos cuantos caracteres gráficos que Vd. se ha definido, quizás pequeños marcianitos. Vd. puede escoger un caracter estandard que no se usa muy frecuentemente, ej.:@ó© y redefínalo.

En una situación de tratamiento de texto Vd. puede desear tener el conjunto de caracteres que no contenga caracteres Ingleses, pero sí un abecedario Griego o Ruso.

Para entender com funciona esto, necesitamos saber como son originalmente almacenados los caracteres. Mirando en el mapa de memoria nos muestra que el conjunto estandard queda almacenado entre las posiciones 46080 y 47104, es decir consumen hasta 1K (1024) bytes. Si hay 128 caracteres, entonces consumen hasta 8 bytes (128x8=1024). Esto le puede hacer imaginar a Vd. un tablero de ajedrez de 8x8 posiciones en el que se almacenan. Sin embargo, esto no es del todo exacto lo que hace el ORIC. En cada byte, sólo los últimos seis bits contienen información de caracteres.

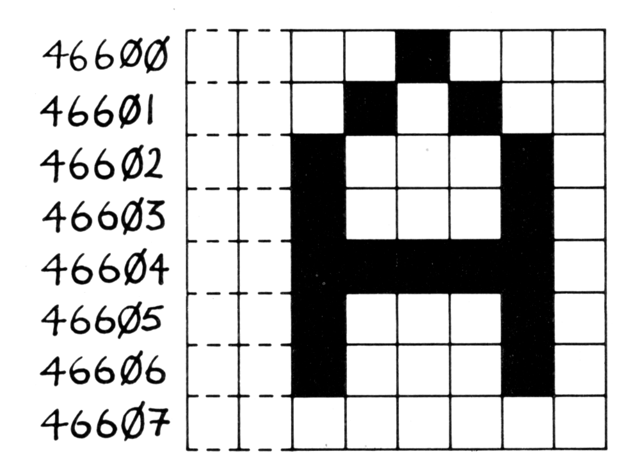
Memory location	Binary Value								Decimal equivalent
466ØØ	0	0	0	0	1	0	0	0	8
466Ø1	0	0	0	1	0	1	0	0	20
466Ø2	0	0	1	0	0	0	1	0	34
466Ø3	0	0	1	0	0	0	1	0	34
466Ø4	0	0	1	1	1	1	1	0	62
466Ø5	0	0	1	0	0	0	1	0	34
466Ø6	0	0	1	0	0	0	1	0	34
466Ø7	0	0	0	0	0	0	0	0	0

Así pues, puede pensarse de que cada caracter ocupa 8 filas de 6 puntos. Cada fila es un byte, de los cuales los últimos 6 bits determina su diseño. Si un bit es 1 entonces la casilla es a "sí" y si es 0 es a "no".

Las casillas a "sí" están en color de primer término, las casillas a "no" están en el fondo.

Para ver todos los contenidos de cualquier posición en memoria, usamos el **PEEK.** Para cambiar los contenidos, usamos **POKE.**

Los caracteres en el conjunto son almacenados en el orden de sus códigos ASCII. El código ASCII para **A** es 65, de manera que el patrón para **A** debería ser almacenado en 46080+(65*8) es decir, 46600 y los siguientes 7 bytes.



Los ceros y unos forman una letra **A.** Es muy fácil para cambiarla. Pruebe esto:

```
POKE   46600, 31
POKE   46601,21
POKE   46602,31
POKE   46603,4
POKE   46604,31
POKE   46605,4
POKE   46606,10
POKE   46607,17
```

Ahora escriba una **A** -¡Vd. ha sido invadido por el ORIC! Ahora todas las **A** salen de esta nueva forma, ya que Vd. ha redefinido la forma en que el ORIC ha de dibujar las **A**.

Tardaría mucho tiempo hacer un **POKE** (PONER) números de uno en uno, así pues que es mejor escribir un programa corto para redefinir caracteres.

A continuación hay un programa espantoso que redefine el conjunto entero de caracteres. Cuando Vd. lo haya entrado, lístelo, entonces ejecútelo, y observe la pantalla. ¡Es una dominación total de marcianos!. La única forma de escapar de ellos es pulsando el botón del reset del ORIC, y Vd. volverá al conjunto de caracteres estandard.

```
5     REM *** INVASION ***
10    FOR X= 46344 TO 47088 STEP 8
20    FOR I= 0 TO 7
30    READ M
40    POKE I+X,M
50    NEXT I
60    RESTORE
70    NEXT X
80    DATA 18,12, 30, 45, 45, 30, 18, 0
```

Aquí hay otro programa más útil. Vd. puede usarlo para redefinir cualquier caracter del teclado, en mayúsculas o minúsculas. Una matriz bidimensional, **Y**, se fija para macenar el nuevo caracter en su gran forma. **C$** es cualquier tecla que se pulse. **A** es el comienzo del conjunto de caracteres estandard en memoria, **D** es la posición del primer byte del caracter escogido en el conjunto de caracteres estandard. La subrutina en la línea 1000 hace un **PEEK** (RECOGE) ese byte del caracter y convierte los contenidos decimales de ese byte en un número binario, es decir, 45 sería convertido en 00101101, y tanto los ceros como los unos son grabados dentro de la matriz **Y**.

La última sección recoge tanto un bloque sólido (128) si la casilla es un 1, o un espacio en blanco (32) si la casilla es un 0 en la pantalla. Esto produce una versión grande del caracter para aparecer en la pantalla. El resto del programa le da al usuario la oportunidad de entrar datos recientes, una línea de una sola vez, y para examinar el resultado en tamaño normal o agrandado.

```
5     REM *** GENERADOR DE CARACTERES ***
10    CLS
20    DIM X(8):DIM Y(8,8)
30    PRINT "POR FAVOR, DIGAME EL CARACTER QUE
      DESEA REDEFINIR"
```

```
  40   GET C$
  50   PRINT C$
  60   C= ASC(C$)
  70   A=46080:D=C*8
  80   GOSUB 1000
  90   PRINT "ENTRA EL DATO"
 100   FOR N=0 TO 7
 110   PRINT "COLUMNA";N;
 120   INPUT X(N)
 130   IF X(N)> 63 OR X(N) <0 THEN 120
 140   POKE (A+D+N),X(N)
 150   NEXT
 160   GOSUB 1000
 200   STOP
1000   REM *** SUBRUTINA ***
1010   FOR N= 0 TO 7
1020   X(N)=PEEK (A+D+N)
1030   FOR M= 0 TO 7
1040   Y(N,M)=INT(X(N)/2 ↑ (7-M)
1050   Z= ((X(N)/2 ↑ (7-M))-Y(N,M))*2 ↑ (7-M)
1060   X(N)=Z+0.0001
1070   IF Y(N,M)=0 THEN POKE 48220+(N*40)+M,32
1080   IF Y(N,M)=1 THEN POKE 48220+(N*40)+M,128
1090   NEXT M
1100   NEXT N
1110   RETURN
```

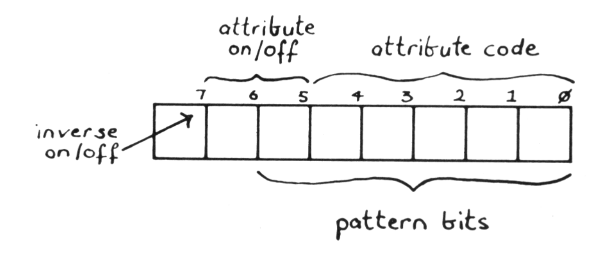
* * * * * * * * *

ATRIBUTOS SERIE

Para enviar información a la pantalla, el ORIC usa atributos serie. Esto significa que un byte enviado a la pantalla puede considerarse como un patrón gráfico o como un atributo controlando el color, destelleando, etc.

La forma en que los bits son agrupados determina cuando se lee un byte como un atributo o no. Si los bits 6+5 son ambos cero, entonces los 5 bits restantes son considerados como un atributo - hay 32. Si los bits 6+5 no son ambos cero, entonces los bits 5 al 0 son leidos como un patrón.

En modo **HIRES**, los bits del 0 al 5 son los bits patrones. En modo **TEXT** y **LORES**, los bits 0 al 6 son los códigos del ASCII. Los códigos de control (los bits 6+5 fijados a 0) se convierten en atributos. El bit 7 controla si el caracter es inverso o no, 1 está a sí, 0 es a no.



Si se fija un atributo, entonces continua hasta el final de la línea, a menos que se cambie. Para ver como los atributos serie pueden ser utilizados para controlar la pantalla en modo **TEXT**, pruebe este programa.

```
10    FOR A=0 TO 255
20    FOR N= 1 TO 24
30    PRINT A;"UN POCO DE TEXTO"
40    NEXT N
50    FOR J= 48042 TO 49002 STEP 40
60    POKE J,A
70    NEXT J,A
```

Las líneas 20 a la 40 llenan la pantalla con texto. Las líneas 50+60 hacen un **POKE** (PONEN) la variable **A** en la pantalla. Debería ver que valores de **A** controlan el color, destelleante, etc. y cuales son impresos en texto normal o inverso. Cuando la A está entre 24 y 31, la pantalla parecerá muy extraña. Esto es porque Vd. está cambiando temporalmente la sincronización. Para más detalles, vea el Apéndice C.

Este programa muestra como los atributos serie pueden controlar el color de caracteres predefinidos en la pantalla **TEXT**.

```
  5   REM ***INVASORES***
 10   GOSUB 1000:CLS
 20   FOR M= 1 TO 20
 30   PAPER INT(RND(1)*4)+4
 40   A=RND(1)*32
 50   ZAP
 60   FOR N= 0 TO 1100 STEP 40
 70   POKE 48039+N+A,1
 80   POKE 48040+N+A,64
 90   POKE 48039+N+A+6,2
100   POKE 48040+N+A+6;64
```

```
 110    POKE 48039+N+A+3,3
 120    POKE 48040+N+A+3,63
 130    SOUND 1,N/2,0
 140    PLAY 1,0,5,5
 150    POKE 48040+N+A,32
 160    POKE 48040+N+A+6,32
 170    POKE 48040+N+A+3,32
 180    NEXT N
 190    EXPLODE
 200    WAIT RND(1)*200+100
 210    NEXT M
1000    REM ***DEFINICION DE CARACTERES***
1010    FOR N= o TO 7
1020    READ X:POKE 46080+(64*8)+N,X
1030    NEXT N
1040    DATA 18, 12, 30, 45, 45, 30, 18, 0,
1050    RETURN
```

Desde este programa Vd. verá que puede tener todos los colores a la vez en primer término de la pantalla, lo mismo que cambiar los colores del fondo. Vd. no tiene que hacer ningún **POKE** en la pantalla -pero lo puede hacer con un **PLOT** (DIBUJAR).

Normalmente lsa columnas protegidas a la izquierda de la pantalla controlan los colores del **INK** (TINTA) y del **PAPER** (PAPEL) para toda la pantalla. Si Vd. pone un atributo en la pantalla, ocupa el cuadro de un caracter y afecta a todos los cuadros de caracteres de su derecha a menos que Vd. ponga otro atributo después. Como el atributo **INK** queda separado del de **PAPER**, no es necesario cancelarlo a no ser que aparezcan otros caracteres a su izquierda.

En modo **HIRES**, es posible tener una resolución de color de 200 líneas por columnas de 40 caracteres. Otra vez, Vd. tiene que poner la posición justo a la izquierda que Vd. desea alterar.

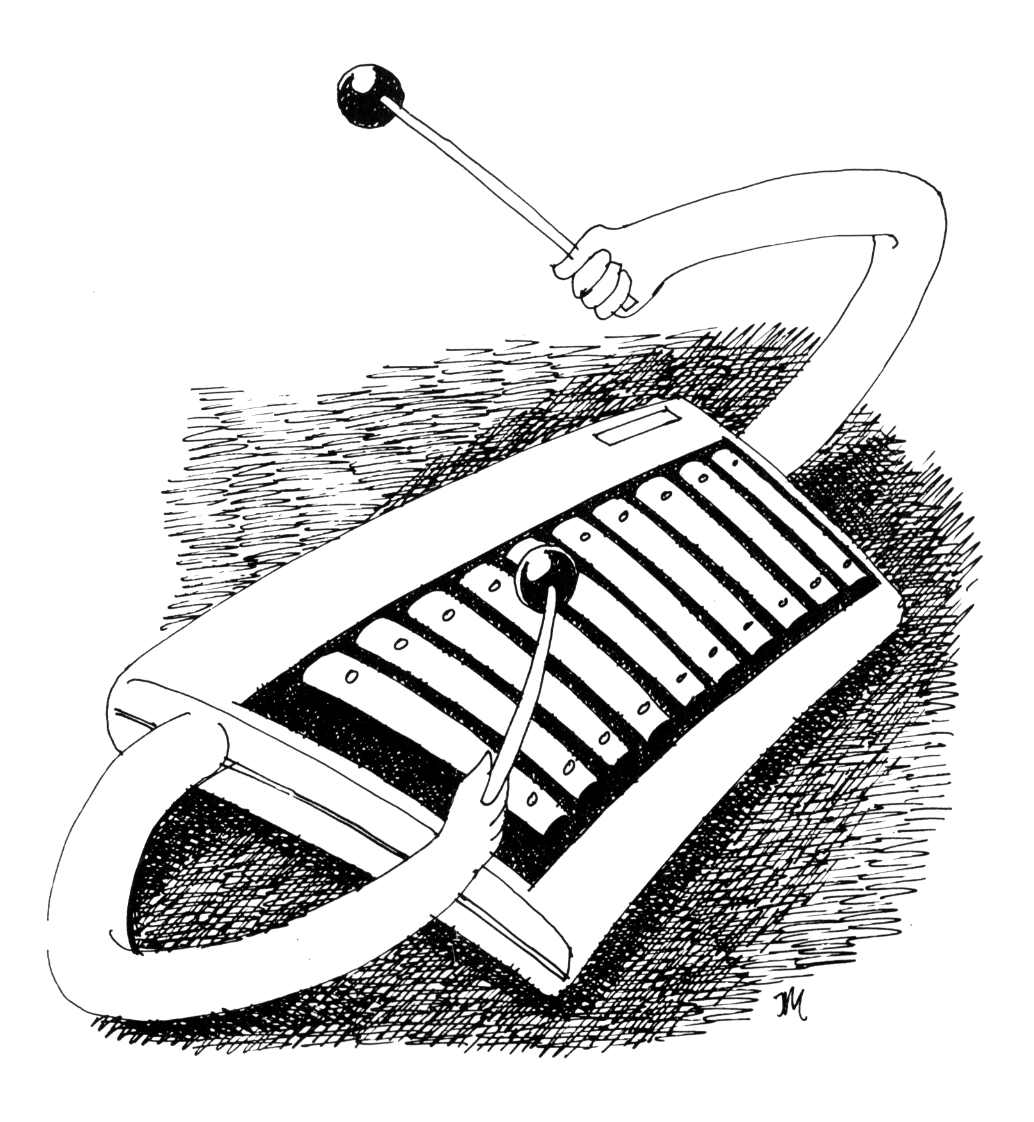
Este programa PONE atributos de fondo en el centro de la pantalla y los de primer término más a la izquierda. Los círculos que están dibujados toman el color de los atributos de acuerdo a su posición en la pantalla.

```
5     REM *** CIRCULO SECCIONADO ***
10    HIRES
20    FOR N=41060 TO 48979 STEP 40
30    POKE N, INT(RND(1)*7)+1
40    POKE N-45, INT(RND(1)*7)+16
50    NEXT N
60    CURSET 120, 100, 3
70    FOR X= 95  TO 1 STEP-1
80    CIRCLE X,1
90    NEXT X
```

El segundo programa muestra como un gráfico puede tener un comienzo multicolor y entonces es sólo un color para una sección ya predefinida. Naturalmente, Vd. puede poner en la pantalla atributos destelleantes o de doble altura usando este método.

```
5     REM ***SENOIDAL COLOR***
10    HIRES
20    FOR N=40960 TO 49079 STEP 40
30    POKE N,INT(RND(1)*7)+1
40    POKE N+100,1
50    NEXT N
60    FOR A= -PI TO STEP 0.02
70    CURSET A*38+120,SIN(A)*99+99,1
80    NEXT A
```

CAPITULO 10
Sonido



10. Sonido

El ORIC contiene algunos comandos de sonido muy sofisticados, usando un chip especial que puede sintetizar 3 tonos diferentes además de un canal de ruido.

Vd. ya ha experimentado con algunos de los sonidos disponibles -cada vez que Vd. pulsa una tecla, el ORIC da un tono alto. Si Vd. pulsa **CTRL** o **RETURN** o cualquier otra tecla de control, Vd. obtendrá un tono bajo. Pruebe pulsando **CTRL** y a la vez pulse **F**. Esto desconectará el sonido de las teclas. Si Vd. pulsa **CTRLF** por segunda vez, volverá el sonido.

¡Ahora un poco de emoción!

Escriba **ZAP** y después **RETURN.** Esto producirá el tono de un silbido como si de una "pistola galáctica Láser" se tratara. Ahora pruebe **PING** -un tono parecido al de una campanilla que puede ser producido pulsando **CTRL** y **G.**

SHOOT simula el parecido del sonido del disparo de una pistola. **EXPLODE** genera una explosión.



Estos son los cuatro sonidos predefinidos que son muy útiles para los juegos de acción. Pueden utilizarse en los programas como comandos BASIC.

```
10    FOR N= 1 TO 10
20    ZAP
30    NEXT N
40    WAIT 15
50    EXPLODE
```

Esto dispara una salva de **ZAPs.**

NOTA: -Vd. debe incluir una pausa, como en la línea 40, para permitir que el sonido termine antes de disparar el siguiente. El rato de espera depende del sonido.

Los principales comandos de sonido son **SOUND, MUSIC** y **PLAY.** En la mayoría de los programas será necesario definir el tipo de sonido con los primeros dos comandos, y para controlar la envolvente el tercero. La envolvente determina la "forma" del sonido, es decir si empieza ruidoso como una guitarra o suave como un órgano. Estos comandos le tomarán bastante tiempo para llegar a familiarizarse con ellos ya que ofrecen la oportunidad de hacer que el ORIC suene como muchos instrumentos ¡lo mismo a los que Vd. le gustaría inventar!. "Ruido blanco" puede añadirse para dar los efectos de bombas, aviones, etc. Las posibilidades sólo están limitadas por su imaginación y no son tan complicadas como parecen al principio.

A continuación hay dos ejemplos de programas cortos que mostrarán algunas de las cosas que Vd. puede hacer.

```
20    MUSIC 1, RND(1)*6,RND(1)*12+1,7
30    WAIT RND(1)*20+5
40    GOTO 5
```

Vd. puede parar la ejecución de este programa escribiendo **CTRL C.** Para saber como funciona vea más adelante los detalles sobre el comando **MUSIC.**

El siguiente programa es un poco más largo, pero le da la oportunidad para usar el ORIC como un instrumento de teclado. Las teclas de la fila de arriba actúan para producir notas un semi tono, empieza en la C, y termina con la B (la tecla "="). Pulsando "/" parará el programa. (Incluya siempre un comando **PLAY 0,0,0,0** al final o la última nota continuará hasta que pulse otra tecla).

```
20    GET A$
30    A=VAL(A$)
40    IF A$="-" THEN A=11
50    IF A$="=" THEN A=12
60    IF A$="/" THEN PLAY 0,0,0,0: STOP
70    IF A= 0 THEN A= 10
80    MUSIC 1,3,A,5
```

90 GOTO 5

La línea 20 espera que se le haga una entrada desde el teclado. La línea 30 lee el valor dentro de la variable A. Si Vd. pulsa un número entonces A será el valor de ese número, si es otra tecla, entonces A será igual a 0. Las líneas 40 a la 60 convierten las teclas restantes a los valores requeridos. La línea 70 evita que la A sea 0. (Esto daría como resultado un mensaje de error ya que el valor de la nota en el comando **MUSIC** no puede aceptar 0 como un parámetro válido).
A continuación verá los detalles de los comandos de sonido.

1. SOUND (Canal, Período, Volumen)

Todos los parámetros deben ser numéricos. Fuera del rango los errores serán detectados.

Canal= 1, 2, ó 3 para canales de tono.
4,5 ó 6 para canales de ruidos

Observe que sólo hay un canal para ruidos, el 4,5 o 6 sólo especifican que canal de tono está mezclado.

Volumen= 1 al 15 niveles de volumen fijados
nivel de volumen de la variable 0 controlado por el comando **PLAY**

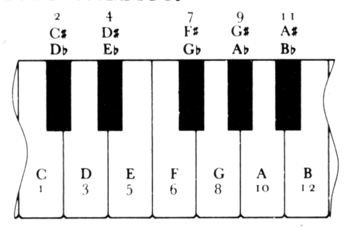
SOUND puede ser usado para producir una gran variedad de sonidos musicales y no musicales. Los canales 1, 2 y 3 producen tonos puros, y 4, 5 y 6 añaden ruidos a cada tono. El valor del período controla el tono, (el nombre se refiere al período de vibración o frecuencia de la nota -no lo confunda por el parámetro de longitud de una nota). A menos de que Vd. esté usando un amplificador externo, probablemente encontrará que los volúmenes 6 ó 7 ¡son suficientemente ruidosos!.

2. MUSIC (Canal, Octava, Nota, Volumen)

Canal= 1,2 ó 3 -canales de tono
Octava= 0 al 6 con 0 dando el tono más suave.
Nota= 1 = C Cualquier otro número producirá un mensaje de error.
2 = C #
3 = D
4 = D #
5 = E
6 = F
7 = F #
8 = G

9 = G ♯
10 = A
11 = A ♯
12 = B

MUSIC ha sido designado para ofrecerle tonos puros, y la altura tonal ha sido fijada para entrar más fácilmente notas de un valor particular, es decir, desde la partitura de la música. Hay tres canales disponibles, y notas, octava (desde la 1 a la 7), y los volúmenes son todos seleccionables.



Si se escoge el nivel cero de volúmen en **SOUND** o **MUSIC**, entonces la salida es dirigida a la sección del comando **PLAY.** Ambos el **SOUND** y **MUSIC** se conectan por el **PLAY.** Observe que la longitud puede ser controlada por las sentencias **WAIT** y el sonido se desconecta por **PLAY 0,0,0,0.**

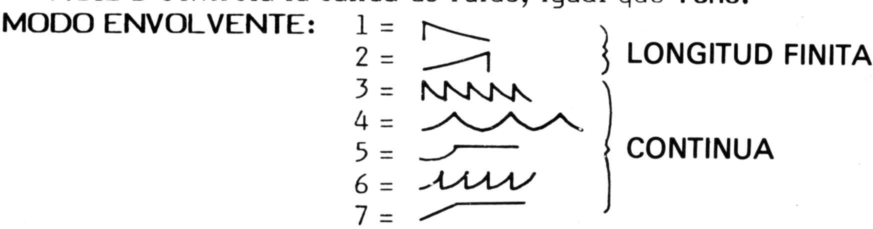
3. PLAY (TONO, RUIDO, MODO ENVOLVENTE, PERIODO ENVOLVENTE)

TONO=

0 = TONO OFF
1 = Canal 1 ON
2 = Canal 2 ON
3 = Canal 1+2 ON
4 = Canal 3 ON
5 = Canal 3+1 ON
6 = Canal 3+2 ON
7 = Canal 3+2+1 ON

RUIDO controla la salida de ruido, igual que **Tono.**

MODO ENVOLVENTE:



11. Grabación de programas en cassette

Cuando Vd. ha perdido mucho tiempo escribiendo un programa largo, es bueno saber que Vd. puede almacenar su programa en alguna parte y grabarlo en el ORIC ú otro ORIC en una situación posterior.

Vd. necesitará un grabador de casettes y un cable de conexión para hacer esto. Tal como habíamos mencionado previamente, los enchufes dependen del tipo de grabador que Vd. tenga. El ORIC tiene un zócalo de 7 patas en la parte de atrás del grabador de casettes. Si éste tiene el control remoto, puede conectarse a más patas. (Si no, no se preocupe -un enchufe DIN de 3 patas debería entrar, pero Vd. tendrá que acordarse de enchufar y desenchufar la máquina, o usar el botón **PAUSE**).

No intente usar un enchufe DIN de 5 patas, pues los pares de patas exteriores están generalmente en corto circuito y no funcionará con el ORIC.

Para salvar un programa, conecte el casette para grabar y escriba

CSAVE "XX"

(XX es el nombre que Vd. le da a su programa y puede tener hasta 17 caracteres de largo e incluye puntos y aparte, guiones, etc.) Cuando Vd. pulse **RETURN**, el programa quedará convertido en señales de sonido y grabado en el casette.

El mensaje

Saving XX

aparecerá en la línea. Cuando el programa haya sido salvado, aparecerá **Ready** en la pantalla.

Para volver a obtener el programa, asegúrese de que el grabador del casette esté conectado correctamente y escriba

CLOAD "XX"

El ORIC buscará a través del casette hasta que encuentre el programa **"XX"** y entonces lo grabará internamente en la memoria.

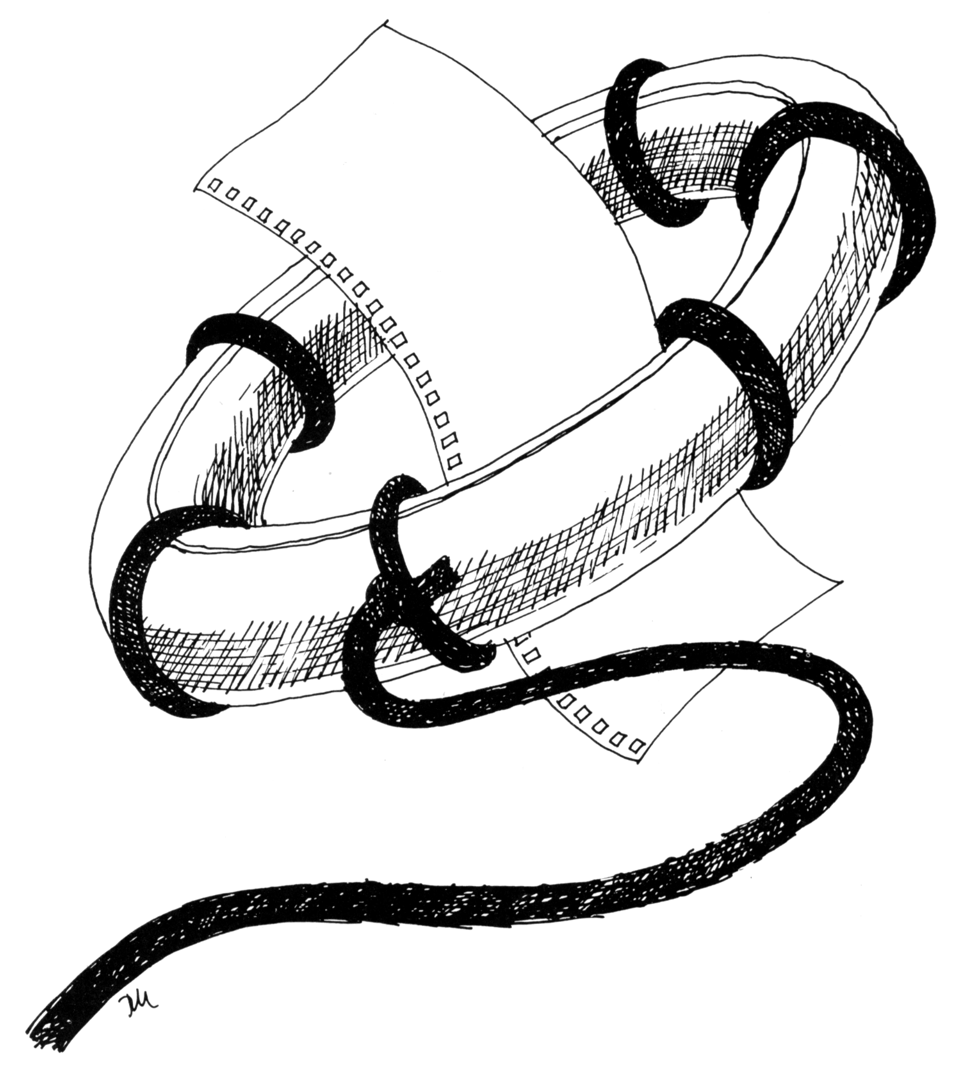
Mientras está buscando, el mensaje

Searching...

aparecerá en la línea. Cuando se ha encontrado el programa deseado, el mensaje cambiará por

```
 60    IF P <0 OR P> 32767 THEN 50
 70    CLS
 80    PRINT "CANAL";T
 90    PRINT "MODO ENVOLVENTE";M
100    PRINT "PERIODO ENVOLVENTE";P
110    MUSIC T, 3, 4, 0
120    PLAY T, 0, M, P
130    PRINT "PULSE RETURN SI EL SONIDO NO PARA"
```

CAPITULO 11
Grabación de programas en cassette



Esto controla la manera de producir el sonido, es decir, repitiendo, rizado, decaimiento, etc.

Período de Envolvente = 0 a 32767

Controla la duración del sonido o nota desde que comienza hasta que desaparece.

Cuando Vd. usa las facilidades del sonido en el ORIC, Vd. puede querer desconectar el golpe seco del teclado pulsando **CTRL** y **F** a la vez. Si no se desconecta ocurrirá que la pulsación de las teclas afectará el sonido que se esté generando.

Este programa ilustra una forma en la cual los valores de las notas, ambos en términos de la altura del tono y la duración, pueden ser contenidos en las sentencias **DATA** y llamados cuando se necesiten durante la ejecución del programa.Se consigue un armónico abriendo los canales 1 y 2 en la sentencia **PLAY.**

```
 10    REM **TONADILLA**
 20    FOR N= 1 TO 11
 30    READ A, 3
 40    MUSIC 2, 3, A, 0
 45    PLAY 3,0,7,2000
 50    WAIT B
 60    PLAY 0,0,0,0
 80    NEXT N
100    DATA 5,30,5,30,7,30,8,75,5,75
110    DATA 8,60,10,30,7,60,5,30,3,30,5,180
```

Aunque **MUSIC** y **SOUND** son fáciles de imaginar en términos de los sonidos que ellos reproducirán, **PLAY** es algo más difícil.

Este programa le permite entrar canales diferentes, 1, 2, y 3, y también alterar los dos parámetros de envolvente, modo y período. De esta manera, Vd. pronto llegará a familiarizarse con todos los comandos de sonido que el ORIC tiene para ofrecer.

```
  5    REM **PRUEBA DE ENVOLVENTE**
 10    INPUT "ENTRA EL CANAL DE TONO (1,2 ó 3)";T
 20    IF T <1 OR T> 3 THEN 10
 30    INPUT "ENTRA EL MODO DE ENVOLVENTE, 0 a 7";M
 40    IF M <0 OR M> 7 THEN 30
 50    INPUT "ENTRA EL PERIODO DE LA ENVOLVENTE, 0 a
       32767";P
```

Loading XX

Si Vd. ha olvidado el nombre del programa, o simplemente desea leer el siguiente programa en el casette, entonces escriba

CLOAD""

Vd. puede comprar casettes de datos especiales de computadora que no son muy largos -C10 ó C15- o puede usar audio casettes de buena calidad. Son preferibles los casettes cortos porque es más fácil localizar un programa.

Una última advertencia -no intente ni grabe en el comienzo de plástico de una cinta. Sus oídos puede que no aprecien la falta de una nota al principio de una melodía, ¡pero el ORIC lo notará aún cuando se trate de un solo byte!.

Lo mismo que al salvar normalmente los programas, el ORIC le permite ser más versátil en el uso de su grabador de casette. Si Vd. hace **CSAVE** en los programas como arriba, éstos quedan grabados a un rango de 2400 baudios (una medida de transferir datos). tenga en cuenta que esta velocidad es sólo posible si el cabezal del casette está limpio y bien colocado, y si Vd. está usando casettes de buena calidad.

Si hay alguna tara en el casette, Vd. puede obtener un mensaje de error.

FILE ERROR - LOAD ABORTED

Si Vd. desea estar completamente seguro de que su trabajo se ha grabado de forma fiable, entonces Vd. tiene que añadir la letra **S** a la instrucción **CSAVE** como verá a continuación, la cual transferirá los datos a la super segura velocidad de 300 baudios.

CSAVE "PROG1",S

Cuando Vd. hace un **CLOAD** a programas cortos, Vd. debe escribir **CLOAD"PROG1",S** o sino el ORIC esperará una grabación rápida.

Si Vd. desea que su programa se ejecute automáticamente una vez ha sido grabado, añada la instrucción **AUTO** a la instrucción **CSAVE.**

CSAVE"PROG1",AUTO

Un programa salvado de esta forma se pondrá en marcha automáticamente al cargarse.

CLOAD "PROG1"

ejecutará inmediatamente después de grabar, pues el mensaje **AUTO** está codificado con el programa en la cinta.

Para salvar bloques de memoria, Vd. necesita saber Address (la dirección) de donde empieza el bloque, y End (donde termina), como se ve a continuación.

CSAVE"PROGMEM",A#400,E#499

Esto salvaría los contenidos de **RAM** de las posiciones .400 a la .499.

Para grabar el bloque, escriba

CLOAD"PROGMEM",A#400,E#499

Como el resto de **RAM** no queda afectado, es posible grabar un nuevo conjunto de caracteres, programas de código de máquina, etc., sin tocar para nada el programa en Basic.

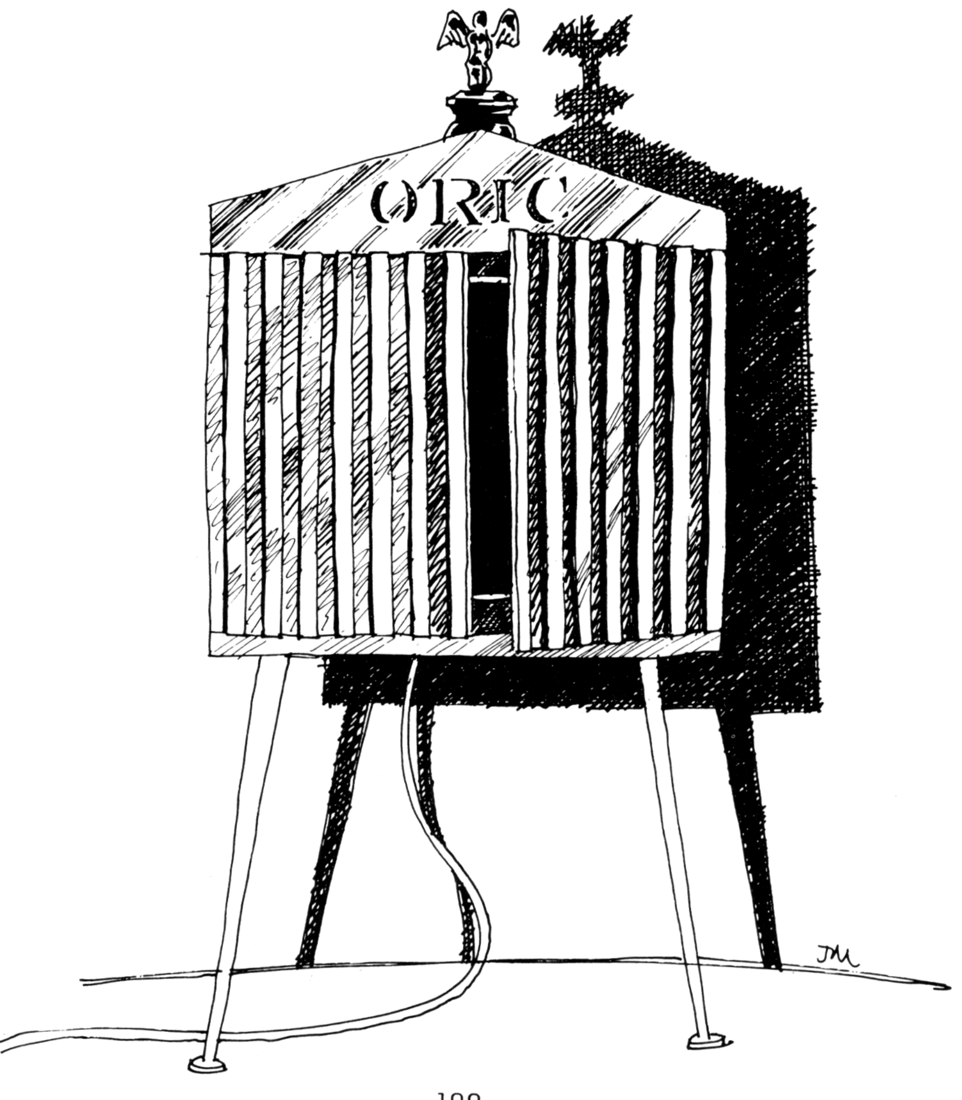
Vd. también puede usar este método para grabar pantallas, y recuperarlas posteriormente. Asegúrese de que si Vd. usa este método, Vd. esté en el mismo modo que necesita la pantalla, ¡o extrañas cosas pueden sucederle!.

Para salvar la pantalla **TEXT** o **LORES**, escriba

CSAVE"PANTALLA BONITA1",A48000,E49119.

Observe que Vd. puede usar números decimales o hexadecimales para las posiciones.

CAPITULO 12
Mejor Basic



12. Mejor Basic

Hasta ahora, Vd. puede haberse sentido bastante confidente usando el ORIC. Probablemente Vd. estará copiando los programas de libros y revistas y quizás empezando a escribir su propios programas originales.

Este capítulo está designado para ayudarle a mejorar la escritura de sus programas y ofrecerle las máximas facilidades que dispone el ORIC.

Los programas de los ejemplos cortos en este manual no gastan mucha memoria. Cuando Vd. enchufa, ya se le ha dicho de cuanta memoria dispone para llenarla con programas. Parte de ella se utilizará para la pantalla **TEXT**, algo más por si Vd. está en modo **HIRES.** Algo más de memoria será usada para almacenar las variables, etc. Para saber lo que queda de memoria, escriba

PRINT FRE (0)

El número de bytes que quedan serán impresos.

Si Vd. escribe programas largos, cada vez que use una variable de cadena, se copia en una parte extra de la memoria. Esto se ve particularmente en los bucles **FOR/NEXT**, o en las subrutinas enlazadas. Si Vd. empieza a quedarse sin memoria, es preferible borrar todas las copias que no hagan falta -despúes de todo, sólo se necesitan las más recientes. Normalmente nosotros lo llamamos "compactar memoria" o como "eliminación de basura". A pesar de que algo de esto se hace de forma automática, Vd. puede forzar esta limpieza de memoria con un comando tal como

240 A=FRE ("")

en la parte necesaria del programa.

Cuando Vd. escribe programas cortos, son fáciles de componer desde el mismo teclado. También es relativamente fácil localizar los problemas, y además Vd. será capaz de ver lo que hace el programa con un simple vistazo al listado.

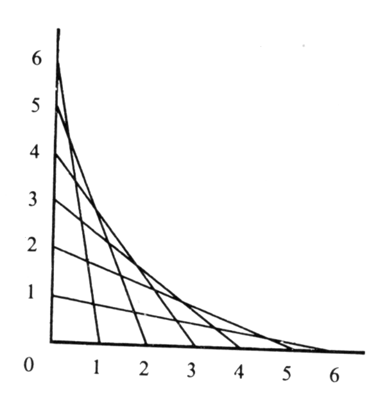
Con programas más largos de 20 líneas, esto conlleva progresivámente más dificultad, y una semana más tarde Vd. puede pensar como lo obtuvo para trabajar en el primer puesto, y para otros puede parecer totalmente incomprensible.

Hay varias formas de hacer su programa más claro para Vd. y para otros. No son reglas difíciles ni rápidas, pero que indudablemente mejoran su programación y la hacen más fácil si Vd. desea pasarla a otros lenguajes como es el Pascal o Forth.

Primero de todo, es una buena idea escribir sus ideas para el programa sobre papel, más que intentar realizarlas directamente desde el teclado. Esto no tiene por que realizarse como un informe tradicional; todo lo contrario, los informes no deberían de tratarse de programas totalmente bien diseñados. Lo único que se necesita es mostrar el orden de los acontecimientos.

Como ejemplo, imagine que le han pedido escribir un programa que mostrará como una serie de líneas rectas se pueden disponer para que formen una curva suave. Esto se llama curva entramada.

Este es el resultado.



El seguimiento de control es como esto:-

RUN → INITIALIZE → INSTRUCTIONS → MAIN PROGRAM → REPEAT OPTION

Las instrucciones han sido incluidas de manera que cualquier otra persona usando el programa sabrá lo que hace y como lo hace funcionar. El hecho de tener en cuenta las necesidades de terceros usuarios se puede llamar diseño "outside-in" (fuera-dentro).

Cada sección del programa se escribirá como un módulo por separado y cada módulo será llamado como una subrutina por separado por el orden que se menciona arriba. Al principio

será un módulo de control para llamar a las subrutinas. Este es el módulo que se escribe primero.

```
2000 REM **MODULO DE CONTROL**
2010 GOSUB 3000    'INICIALIZACION'
2030 GOSUB 5000    'PROGRAMA PRINCIPAL
2040 STOP
```

¡Ahora el programa está escrito más correctamente! Todo lo que es necesario es llenar las subrutinas. En la práctica, Vd. se puede encontrar con subrutinas que ya ha escrito y que le son útiles. Es bastante útil construirse de esto una librería.

Hasta aquí, Vd. encontrará útil usar la computadora para probar los módulos. Estos serán introducidos uno cada vez.

* * * * * * * * * *

INICIALIZACION

Cuando Vd. enchufa el ORIC, todas las variables están fijadas a cero. Esto significa que Vd. no necesita dar a las variables valores iniciales si es que en el programa necesitan el valor de cero, aunque es una buena idea usar nombres y letras que le recuerdan para que están, es decir:-

LONG= 160 (longitud del cuadrado en pixels)

Tenga cuidado de no tener dos variables que empiecen con las dos letras iguales. También puede evitar las palabras reservadas, como **KEY** o **LET** que ya están en Basic.

Si Vd. pone sentencias en **REM** al principio de las subrutinas y cerca de las variables, podrá ver lo que pueden hacer. O sea:-

I= RND(1)*7:REM ** ESCOJE COLOR DE LETRA ALEATORIAMENTE **

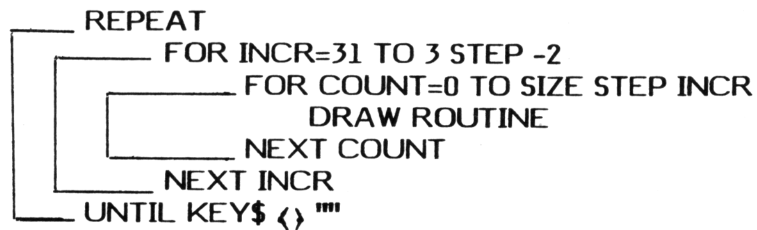
* * * * * * * * * *

MODULO PRINCIPAL

Cuando Vd. alcanza la sección principal del programa, entonces si contiene gráficos es útil apuntar sobre papel la pantalla acabada y trabajar con los valores los cuales guardan las líneas con los márgenes de la pantalla.

¿Cómo se diseñará la pantalla? Es posible definir cada línea en la pantalla de una sola vez, pero esto produce pérdida de memoria y creación. Si Vd. conoce la posición de comienzo de las líneas y la medida, es mejor usar los bucles **FOR/NEXT.**

En este programa, es posible usar dos bucles **FOR/NEXT** anidados dentro de cada uno.



Estos bucles están contenidos dentro de un bucle **REPEAT....UNTIL** que controla la opción de repetición. Observe que estos bucles no se solapan, y que Vd. puede entrar y salir de cada bucle sólo por su camino correcto.

Existe otra característica del ORIC que puede ser muy útil si se usa aquí. Se llama sangrado de programas. Generalmente, los espacios en blanco situados al principio el ORIC no los cuenta, así pues si Vd. entra

```
10    PRINT A
```

el listado mostrará

```
10 PRINT A
```

Sin embargo, si Vd. entra al principio un punto y coma, no habrá ninguna diferencia a la hora de ejecutar el programa, pero los espacios en blancos después del punto y coma permanecerán allí.

Usando esta característica, Vd. puede intentar todas las estructuras del bucle en su programa. A pesar de que esto le hace perder más tiempo, los programas serán más fáciles de entender.

Varios de los programas de este manual sirven para hacer claras sus estructuras. Naturalmente funcionarán con los puntos y comas

Como ya ha sido mencionado anteriormente, es normal numerar las líneas del programa de diez en diez para poder añadir posteriormente más líneas.

Para completar su programa, necesita un título, lo mismo que su nombre y la fecha en la cual Vd. terminó esta versión en particular. Esto le ayuda a saber lo que el programa intenta hacer y permanece cuando Vd. lo repasa.

Más adelante Vd. puede adquirir una impresora. Los listados que se obtienen por la impresora son más pulidos, y por lo tanto más fáciles de descifrar.

El diseño y la construcción de un programa en esta forma se llama programación "top down".

La programación "top down" produce programas claramente estructurados. También implica un escaso uso de las sentencias **GOTO**. Los programadores suelen caer en la tentación de utilizar indiscriminádamente la instrucción **GOTO**, produciendo programas sumamente liados.

```
10    PRINT "ENTRE LA FECHA"
20    INPUT A
30    IF A=30 THEN GOTO 50
40    GOTO 10
50    GOTO 70
60    PRINT A;"ERA MAS DE 100":STOP
70    PRINT A;"ERA EL ULTIMO NUMERO"
80    IF A> 100 THEN GOTO 60
90    END
```

Esto es tan solo un ejemplo, pero muestra como los programas mal estructurados son muy confusos. El uso de **REPEAT/UNTIL**, **FOR/NEXT** y **IF/THEN...ELSE** le evitarán que sus programas sean enrevesados.

A continuación tenemos el programa de curva entramada completo. No pretende ser ninguna maravilla, sino tan solo un ejemplo de utilización de los bucles.

```
1000    :REM **CURVA ENTRAMADA**
1010    :REM
```

```
1020 :REM **COPYRIGHT A.J.S.**
1030 :REM
1040 :REM **1/1/83**
1050 :REM
2000 :REM **MODULO DE CONTROL**
2010 :GOSUB 3000    'INICIALIZACION
2020 :GOSUB 4000    'INSTRUCCIONES
2030 :GOSUB 5000    'PROGRAMA PRINCIPAL
2040 :STOP
2050 :REM
3000 :REM **INICIALIZACION**
3010 :REM
3020 :INK1:PAPER 4
3030 :SIZE = 160
3040 :TEXT
3050 :RETURN
3060 :REM
4000 :REM **INTSRUCCIONES**
4010 :REM
4020 :CLS
4025 :B$ = "L     ****ESPIROGRAFO****"
4030 :PRINT CHR$(27);B$
4032 :PRINT
4034 :PRINT
4035 :PRINT
4036 :PRINT
4040 :PRINT "Este programa dibujará líneas desde"
4045 :PRINT "cada lado de un cuadrado a los lados"
4055 :PRINT "adyacentes. La medida es la distancia"
4060 :PRINT "de cada línea con la de al lado."
4065 :PRINT "se redibujará dentro del paso"
4070 :PRINT "dividido entre dos"
4080 :PRINT
4090 :PRINT "Pulse RETURN para empezar"
4100 :GET AS
4110 :RETURN
4120 :REM
5000 :REM **PROGRAMA PRINCIPAL**
```

```
5010 :REM
5020 :REPEAT
5030 :      FOR INCR = 31 TO 3 STEP -2
5040 :      HIRES
5050 :            PRINT "STEP"INCR
5060 :            INK (INCR/6)+1
5070 :            FOR COUNT = 0 TO SIZE STEP INCR
5080 :                 CURSET 180-COUNT,10,3
5090 :                 DRAW COUNT,SIZE-COUNT,1
5100 :                 CURSET 20,COUNT+10,3
5110 :                 DRAW COUNT,SIZE,-COUNT,1
5120 :                 CURSET COUNT+20,170,3
5130 :                 DRAW SIZE-COUNT,-COUNT,1
5140 :                 CURSET 20,COUNT + 10,3
5150 :                 DRAW SIZE-COUNT,-COUNT,1
5160 :            NEXT COUNT
5170 :            WAIT 100
5180 :      NEXT INCR
5190 :PRINT "PULSE CUALQUIER TECLA PARA
     PARAR":WAIT 500
5200 :UNTIL KEY$ <> ""
5210 :RETURN
```

Los programas estructurados tienen sus incovenientes y Vd. tendrá que decidir cuando se malgasta demasiada memoria. Algunas personas guardan a parte una copia del programa master, junto con las sentencias **REM**, y las explicaciones de como funciona, mientras que las copias de trabajo tienen las **REMS** eliminadas. Siempre es útil acudir a la copia master si Vd. necesita cambiar el programa en el futuro.

Ventajas de un diseño estructurado.

1. El flujo de control es fácil de seguir.
2. Las estructuras entrarán perfectamente en módulos separados.
3. El número de errores será reducido, y esos errores que quedan sueltos quedarán eliminados muy fácilmente.
4. Las ideas interesantes a este desarrollo le harán aprender más fácilmente otros lenguajes y sus aplicaciones.

Desventajas.

1. Los programas estructurados pueden usar más memoria.
2. La velocidad del programa puede ser reducida.
3. El hardware no puede ser usado de la manera más económica.
4. ¡Es más difícil aprender a escribir programas bien estructurados que recoger hábitos funestos!

Atrapar errores.
Escribiendo programas estructurados puede convertirse en un mejor programador, pero no hay ninguna garantía de que su programa sea tan bueno como debería de ser. Cuando Vd. compruebe su pieza de resistencia, entonces si está diseñada para ser usada por otros es importante considerar todas las cosas absurdas que puedan decidir entrar.

Si Vd. pregunta un número para ser entrado, ¿que ocurre si un usuario entra "dos" en lugar de "2"? ¿Qué ocurre si pulsan **RETURN** sin entrar nada?

Afortunadamente, el ORIC es para usuarios errantes. En primer lugar -una string que ha sido entrada en lugar de un número - **REDO FROM START** se imprimirá hasta que se entre un elemento. En segundo lugar, el ORIC esperará hasta que algo haya sido entrado. Si se pulsa constantemente la tecla **RETURN**, entonces reaparecerá constantemente el signo del interrogante.

Vd. puede facilitar sus exigencias más facilmente al usuario para que lo pueda entender. Si Vd. revisa el código ASCII de entrada, o el valor de un número, entonces un mensaje diciéndole al usuario lo que tiene que entrar puede realizarse en su programa.

```
10   INPUT "ENTRAR EL AÑO";A$
20   IF VAL(A$) < 1900 OR VAL(A$) > 1985 THEN print 2entre
     1900 y 1985!"
30   PRINT "GRACIAS"
```

Si Vd. dibuja una línea o mueve un caracter, es importante no salirse de los bordes de la pantalla o imprimirlo en una de las columnas protegidas. Vd. puede tener en cuenta esto, pero no otro usuario.

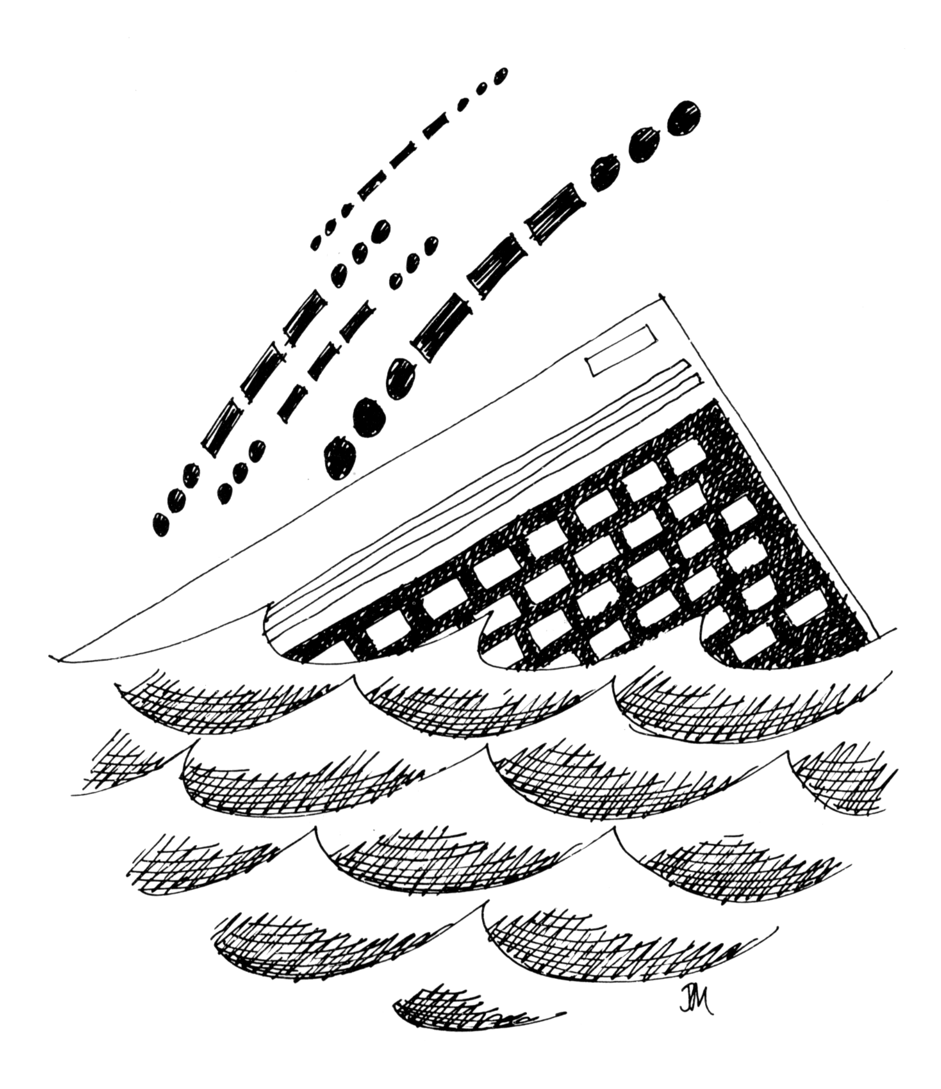
Si A es la posición horizontal y B la posición vertical en la pantalla **HIRES**, entonces algo como esto puede ayudar

```
IF A > 238 THEN A = 238
IF A < 1 THEN A = 1
IF B > 198 THEN B = 198
IF B < 1 THEN B = 1
```

Esto es útil suponiendo que el caracter no es mayor que dos pixels. Cambia los valores de acuerdo con tus necesidades.

Vd. tiene que imaginar las peores cosas que le podrían hacer a su programa, y que alguien, en algún momento, lo podrá romper. No hay nada mejor que un programa perfecto, pero en cierta manera siempre hay la posibilidad de caer en algún error. Entonces Vd. tiene que hacer un programa bien construido.

CAPITULO 13
Programa en código máquina



13. Programa código máquina

Se ha mencionado previamente que las computadoras todavía no entienden los comandos ingleses normales, ya que son ambiguos e idiosincráticos. Lo mejor que pueden es interpretar un lenguaje como el Basic que facilita, hasta cierto punto, un subconjunto limitado de inglés.

Los lenguajes de las computadores pueden ser vistos como jerarquías -lenguajes de alto nivel- y el código binario del lenguaje de máquina -bajo nivel- al final.

El lenguaje de alto nivel que le oferece el ORIC es el Basic. El chip que traduce el Basic en código de máquina se llama ROM del Basic. ROM significa Memoria de Lectura Solo, y el interpretador del programa que contiene se fija durante la fabricación, y no puede cambiarse. Si Vd. tiene un modelo de 48K, realmente contiene 64K de RAM (Memoria de Acceso al Azar que puede ser cambiada, y que generalmente contiene sus programas).

Si Vd. compra unidades de disco para más almacenamiento de memoria y para que sea más rápido, el ROM interno es enmascarado, dejando alrededor de 64K de memoria interna. De esta manera, otros lenguajes de alto nivel, como el Forth, Pascal, Logo, Prolog y Lisp podrían utilizarse en el ORIC.

Con todo este potencial para lenguajes de alto nivel, que son más claros de entender, Vd. puede pensar por que todo el mundo debería molestarse con los códigos de máquina. Después de todo, se estima que por lo menos tarda diez veces para escribir como su programa equivalente en un lenguaje de alto nivel. Vd. no obtiene mensajes útiles de errores, las faltas son más difíciles de descubrir, y el código de máquina es difícil para documentar y de entender - ¿así pues por qué intentar aprenderlo?.

Entendiendo el código de máquina le ayudará a entender el funcionamiento de las computadoras, y los programas eficientes en código de máquina son ejecutados mucho más deprisa que cualquier lenguaje de alto nivel. Si Vd. se imagina hablando a un alemán, que a la vez tiene que traducir sus instrucciones a un italiano antes de realizar una operación, ¡Vd. verá la ventaja de poder hablar directamente al italiano en su propio lenguaje!.

Hay varias formas en que los lenguajes de máquina se hacen más comprensibles. Primero, es normal escribir en hexadecimal - una página de binarios pronto se convierte en una masa de ceros y unos, y los números decimales no le muestran lo que está ocurriendo a nivel de bytes.

Esto es porque el ORIC facilita entrar los números en forma decimal o hexadecimal, en lugar de perder el tiempo haciendo conversiones de base. También le facilita un código que es más fácil para leer.

En el corazón del ORIC lo más importante es el chip, la unidad central del procesador.

Todas las computadoras necesitan un C.P.U. pero no todas necesitan usar el mismo modelo. Cualquier otra computadora que use el mismo C.P.U. puede, con los límites de la computadora, usar el mismo programa de código de máquina. El ORIC usa un procesador 6502 de la Corporación Internacional Rockwell. Vd. se puede encontrar con otros procesadores como el 6800, 6809, Z80 y 8080. Todos operan de diferentes maneras, así pues entienden diferentes.instrucciones.

Internamente, el procesador maneja los números, almacenados como dígitos binarios de 8 bits. Los números son grabados en diferentes posiciones de memoria y tratados como instrucciones o datos.

Por ejemplo, si el 6502 recibe el número 10101001, entiende que esto es una instrucción: grabe un registro, o un espacio de memoria especial llama al acumulador con el siguiente número recibido. El 6502 sólo entiende los números de 8 bits. El ORIC le permite entrar números decimales o hexadecimales, y los convierte en sus equivalentes binarios antes de enviarlos al 6502.

Para hacerlo más fácil de entender, a cada instrucción 6502 se le da un nombre corto, así pues en el ejemplo anterior, 10101001, o 169 (decimal) o#A9 (hex) se conoce como LDA (Acumulador de Grabación). A estos nombres se les llama mnemonics. Para que Vd. se introduca en el uso de códigos de máquina del ORIC, hay una sección en los apéndices sobre los mnemonics 6502.

Aunque entrando A9 en el ORIC sería entendido, el mnemonic no sería reconocido, y para hacerlo más claro, se tendría que añadir como una sentencia **REM**, es decir

```
100 DATA# A9,# 20 'LDA# 20
```

Para ayudarle a entrar programas en código de máquina, es posible tener un programa corto que le permitirá entrar mnemonics directamente, lo mismo que las variables, datos y direcciones, etc., y esto será codificado en lenguaje de máquina. A esto se le llama ensamblar un programa. El usuario entra los mnemonics (la "base" del programa) y el ensamblador traduce esto a código de máquina o programa del "objeto". Para desemsamblar

un programa, funciona a la inversa.

* * * * * * * * *

PEEK y POKE

¿Cómo podemos decir lo que se almacena en las posiciones de memoria?. Vd. puede recordar el **PEEK** y **POKE** en capítulos anteriores. Escriba:-

PRINT PEEK (48225)

Esto es mirando en la parte de memoria que se usa para almacenar la pantalla **TEXT.** El número devuelto es el decimal equivalente del número binario almacenado en esa posición. Como esta posición es mapeada en el área de la pantalla, el número es el código ASCII para el caracter en esa posición. Para cambiar ese valor, escriba:-

POKE 48225,128

Vd. debería ser capaz de ver que posición en su pantalla controla 48225, pues será llenado con el caracter representado por ASCII 128, es decir un bloque sólido.

Si Vd. mira el mapa de memoria en el apéndice, verá que es almacenado en diferentes posiciones. Vd. puede intentarlo usando **POKE** para poner caracteres en la pantalla **TEXT** y **HIRES.**

Esta técnica ya ha sido usada en el capítulo de gráficos. No es una buena idea intentar hacer un **POKE** en las páginas 0 a la 3 (0 al 1024 en decimal). Naturalmente Vd. puede practicar, pues no dañará al ORIC, no importa lo que entre.

* * * * * * * * *

DIRECCIONES

Quizás ha pensado como el ORIC puede almacenar números mayores que 255, particularmente porque hay 65536 diferentes posiciones de memoria. Las direcciones son almacenadas como dos números de bytes, es decir

	1er byte	2º byte
128 es almacenado como	10000000	00000000

Si el número es mayor que 255, el segundo byte contiene el número de 256 en el número.

es decir 258 es almacenado como

1er byte		2º byte
00000010		00000001
2	+	(1 x 256)

Le puede parecer extraño que el orden de bytes más bajo venga primero, pero este es el orden en el cual el procesador lo descodifica. Así pues, si Vd. sabe que un número es almacenado en las posiciones 20345 y 20346, para calcular el número almacenado necesitaría escribir:

PRINT PEEK (20345) + (PEEK(20346)*256)

El ORIC le ahorra de hacer esto. La instrucción

PRINT DEEK (20345)

hará el mismo trabajo que la línea de arriba. **DEEK** viene de **D**oble **PEEK.** Si Vd. quería cambiar el número contenido en estas dos posiciones, Vd. escribiría

DOKE 20345,N

y **N** se convertiría en un número de dos bytes.

* * * * * * * * * *

Algunas veces es útil poder usar rutinas cortas de código de máquina, aunque el resto de su programa puede estar escrito en Basic. Esta pueden ser totalmente originales, o puede cogerse rutinas que ya están escritas en ROM por el **PEEK.**

Hay varios comandos en el BASIC del ORIC que le permiten hacer esto.

CALLX, donde **X** es una dirección en memoria, tranfiere el control a la dirección especificada y empieza la rutina del código de máquina que está allí. Para volver al Basic se hace cuando la rutina alcanza un **RTS.** (**R**e**T**orna de la **S**ubrutina).

Otra manera de acceder a la informción desde una rutina de un código de máquina es usando **DEF USR** y **PRINT USR(0).** La rutina está escrita en código de máquina y la dirección que empieza es entrada por **DEF USR** = dirección de comienzo.

Si **PRINT USR(0)** es entrado ahora, el resultado de la rutina es sacado desde el acumulador del punto flotante e impreso. A continuación hay un ejemplo de como puede ser usado:

```
  5   REM ***RAD/DEG CONSTANT***
 10   FOR DISP = 0 TO 12
 20   :    READ DTA
 30   :    POKE #400+DISP,DTA
 40   NEXT DISP
100   DATA # A9, # 07                'LDA #CON57;LO
110   DATA # AO, # 04                'LDA # CON57;HI
120   DATA #4C, #73, #DE             'JMP MOVFM;FOLAT.
130   DATA#86, #65, 2E, #EO, #D8 'CONSTANT180/PI
```

Ejecute el programa, entonces escriba **DEF USR** = #400. En cualquier momento, **PRINT USR (0)** imprimirá la constante conversión para los radianes a grados.

Aquí hay una explicación más detallada de como funciona. El valor que está entre comillas después de la función **USR** es realmente trasladado al acumulador del punto flotante y un **JSR** a la posición #21 es realizado. La posición #21 a la #23 debe contener un **JMP** al principio de la posición de la subrutina del lenguaje de máquina. El valor retornado por la función se coloca en el acumulador del punto flotante.

Para obtener un entero de 2 bytes del valor en el acumulador del punto flotante, la subrutina debería hacer un **JSR** al #D867. Hasta el retorno, el valor del entero estará en las posiciones #34 (byte de orden alto) y #33 (byte en orden inferior).

Si Vd. desea convertir el resultado de un enetero a su equivalente punto flotante, de manera que la función pueda devolver ese valor, el entero de dos bytes debe colocarse en los registros A (byte de orden superior) e Y(byte con orden inferior). Si se hace un **JSR** a # D8D5, entonces hasta el retorno, el valor del punto flotante habrá sido grabado en el acumulador del punto flotante.

Hay otras dos operaciones útiles que el ORIC puede realizar.

! puede definirse como un comando que todavía no existe en el Basic del ORIC.

&(X) (donde X = 0 a # FFFF)

puede ser definido como una función que todavía no existe en el Basic del ORIC.

Las rutinas tienen que ser escritas en código de máquina y grabadas en una posición determinada en la memoria. La dirección de comienzo se graba de la siguiente manera:

DOKE . **2F5,** dirección - empieza la dirección de la rutina !.
DOKE . 2FC, dirección - empieza la dirección de la rutina &.
Para definir **!** para que signifique **PRINT AT** escriba:-

```
  5   REM*****PROGRAMA PARA LA EXTENSION CMD
      PARA 'PRINT . X,Y;JJJJ'
 10   REPEAT
 20   READ DTA
 30   POKE . 400+CL,DTA
 40   CL=CL+1
 50   UNTIL DATA= .FF         :REM END OF PROG.
100   DATA . 20, .96, .D9     :REM JSR GTVALS
110   DATA . AC, . F8, .0/2   :REM LDY GCOL
120   DATA . C8               :REM INY
130   DATA . 8C, .69, .02     :REM STY CURCOL
140   DATA . A5, .1F          :REM LDA GCL
150   DATA .A4, .20           :REM LDY GHC
160   DATA .85, .12           :REM STA CURBAS
170   DATA .84, .13           :REM STY CURBAS+1
180   DATA .A9, .3B           :REM LDA . ';'
190   DATA .20, .DB, .CF      :REM JSR SYNCHR
200   DATA .4C, .61, .CB      :REM JMP PRINT
210   DATA . FF
220   DOKE . 2F5, . 400
```

Si Vd. escribe:
!X,Y;"ORIC"

entonces la palabra ORIC se imprimirá en las coordenadas X,Y en la pantalla.

Para definir **&** para que indique RETORNA LA POSICION DEL CURSOR VERTICAL:-

```
  5   REM **EXTENSION CMD VERT/CURS/POS
 10   FOR N= 0 TO 5
 20   READ DTA
 30   POKE . 400+N,DTA
 40   NEXT
 50   DOKE . 2FC, .400
 60   DATA .AC, .68, .02
 70   DATA .4C, .FD, .D3
```

Es útil saber en que línea el cursor está si Vd. necesita usar la característica de doble altura, pues se le ocupará la línea de debajo.

Una línea en un programa idónea para proteger contra esto sería

```
500 IF & (0)/2 <> INT(&(0)/2) THEN PRINT
```

Esto moverá el cursor hacia abajo a una línea.

Parte del contenido de la página 4 (posiciones # 0400 a la # 0420) ha sido reservado para sus propias rutinas en código de máquina. Cualquier cosa en memoria puede ser usado para programas más largos, pero un programa en Basic puede ser reescrito si ocupa el espacio de los PROGRAMAS DEL USUARIO en RAM.

Para reservar memoria para los programas en código de máquina, el tope del área para el usuario puede ser bajada. Para encontrar esta posición entre

```
PRINT DEEK (#A6)
```

Trabaje con tantos bytes como necesite para su programa, añada un número pequeño para seguridad (a menos que Vd. esté realmente empujado por la memoria), y saque el total del número que Vd. encontró anteriormente. Entonces entre

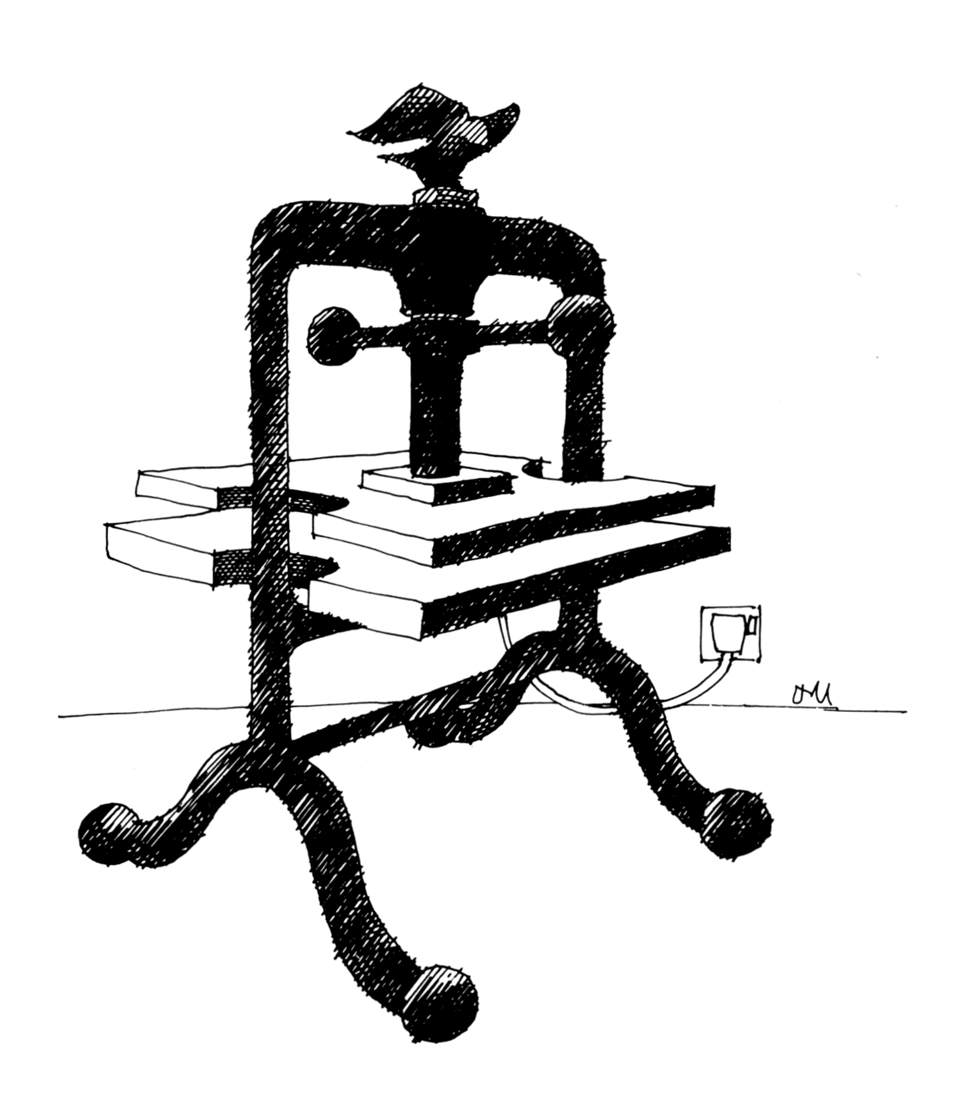
```
HIMEM X
```

donde X es el nuevo tope de memoria para el usuario que Vd. justo acaba de calcular.

Si Vd. desea saber más sobre el uso del código de máquina 6502, entonces encontrará algunos libros que hablan sobre ello con más detalle.

Los más útiles están escritos por Rodnay Zaks (publicado por Sybex) y Lance Leventhal (publicado por Osborne McGraw-Hill). Otra información y libros pueden obtenerse del mismo Internacional Rockwell.

CAPITULO 14
Utilización de impresora



14. Utilización de impresora

El Oric puede ser utilizado como cualquier impresora que tenga una interface Centronics. Lo mismo que la impresora, Vd. necèsitará un cable de conexión. La conexión de la impresora está situada en la parte de atrás, cerca del de expansión.

La impresora debería encenderse antes de ser enchufada. Si todo funciona correctamente la impresora se colocará en la posición de empezar tan pronto como tenga corriente. A parte del libro de instrucciones de la impresora, Vd. encontrará útil la siguiente información.

Una vez la impresora ya esté conectada, ejecute el siguiente programa.

```
10    REM **PRINTER TEST**
20    FOR N=0 TO 255
30    LPRINT N,CHR$(N)
40    NEXT N
```

Cuando haya entrado el programa escriba

LLIST

Esto listará el programa en la impresora en lugar de la pantalla. Si no hace ningún listado, pero aparecen caracteres en Japonés o gráficos, consulte el manual de la impresora y cambie el caracter fijado en la impresora. Cuando el listado sea satisfactorio ejecute el programa.

Como puede ver, consiste de un solo bucle que hará un **LPRINT** (impresión por impresora) al número seguido por el caracter Ascii que representa el código. Esto le mostrará el conjunto de caracteres disponibles y, también le mostrará que números son leídos como códigos de control en la impresora.

Aunque hasta cierto punto están estandarizados, no todas las impresoras responden de la misma forma. Los códigos de control determinan las operaciones como el avance de línea, retorno de carro, papel, alimentador de hojas de papel, tamaño del caracter, etc.

Por ejemplo, las impresoras C.ITOH imprimirán estandard, caracteres comprimidos o expandidos. Sì Vd. fuera a escribir

LPRINT CHR$(14)

entonces los caracteres subsiguientes en la impresora serían de doble tamaño -muy útile para títulos, etc.

LPRINT CHR$(15)

resultaría un retorno a la impresión normal.

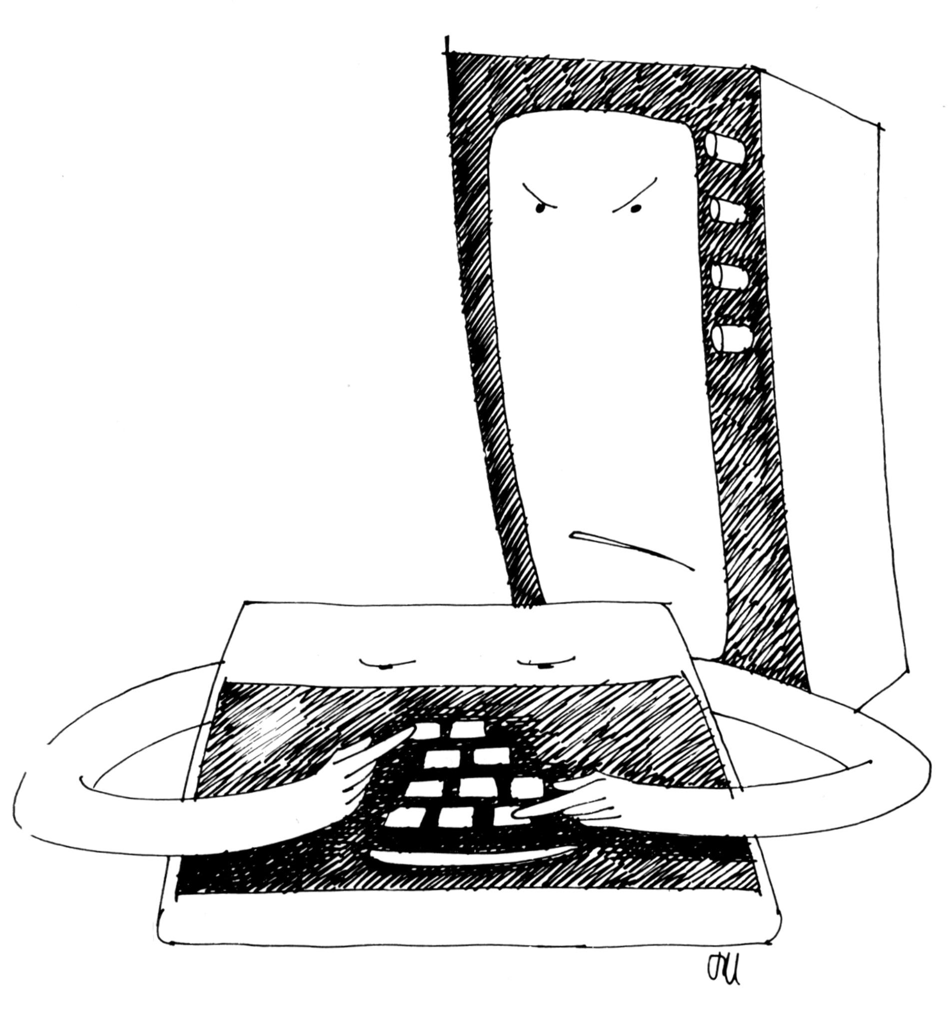
LPRINT CHR$(12)

pasará a la siguiente página. (El papel avanza por el pie de la página). Estos códigos son útiles para saber que pueden ser incorporados en sus programas.

Muchas impresoras que tienen caracteres gráficos también pasarán de la pantalla a la impresora que es una manera de obtener permanentemente una grabación de dibujos, etc.

Probablemente Vd. encontrará una impresora más útil para producir listados, de manera que Vd. puede ver la estructura de un programa completo. También son esenciales para obtener copias de correo electrónico.

CAPITULO 15
Basic del Oric



15. Basic del ORIC

COMANDO	EFECTO	EJEMPLO
ABS	Retorna el valor absoluto	**ABS(-4)es 4** **ABS(4)es 4**
ASC	Retorna el código ASCII del primer caracter en cadena. Vea el apéndice.	**A=ASC(N$)**
ATN	Retorna arcotangente en radianes.	**Z=ATN(Y/4)**
CALL	Transfiere control a la rutina en código de máquina que comienza en la dirección X. Cuando alcanza un RTS retorna al Basic.	**CALL X**
CHAR	Dibuja un caracter en la actual posición del cursor - la parte izquierda del caracter está en la posición del cursor. X es el código del SCII (32-127), S es 0, caracter estandard ó 1, caracter alterno. FB es 0 a 3 (vea abajo).	**CHARX,S,FB**
CHR$	Retorna el caracter ACII que corresponde al valor (32-128).	**CHR$(valor)**
CIRCLE	Dibuja un círculo centrado en la actual posición del cursor. Ninguna parte del círculo puede dejar la pantalla. R es el radio (1-119) FB es 0 a 3(ver abajo).	**CIRCLE R,FB**
CLEAR	Fija las variables a 0, y las strings a nulas (vacías).	**CLEAR**
CLS	Borra la salida en pantalla	**CLS**
CLOAD	Graba el nombre del fichero XX desde la cinta. Para comandos adicionales de cintas, vea el Capítulo 11.	**CLOAD"XX"**
CONT	Continua la ejecución del programa después de la parada.	**CONT**

NOTA:-Códigos FB.

FB es el valor del primer/segundo plano

0	Segundo plano	2	Inverso
1	Primer plano	3	Nulo (no hace nada).

COS	Retorna el coseno del ángulo N (N debe estar en radianes).	**A=COS(N)**
CURMOV	Fija el cursor a una nueva posición. X,Y son relativos a la antigua posición.FB es 0-3 (ver abajo).	**CURMOV X, Y,FB**
CURSET	Fija el cursor a la posición absoluta de X,Y. Nota:-la posición final de X debe ser 0 a 239, e Y 0 a 199 en todos los comandos de gráficos. FB es 0-3.	**CURSET X,Y, FB**
CSAVE	Salva el nombre del fichero XX a la cinta.	
DATA	Almacena una lista de datos que pueden ser leídos dentro de variables. Los espacios directivos se perderán a no ser que se encierren entre comillas.	**DATA 1,2 BATH " ANGIE"**
DEEK	Retorna los contenidos del byte más 256 veces de los contenidos del siguiente byte.	**?DEEK(45610)**
DEF FN	Define funciones numéricas.	**DEF FN A(Z)= Z+4**
DEF USR	Define el comienzo de la rutina USR.	**DEF USR=# 400**
DIM	Dimensiona las matrices. (Las matrices son predimensionadas a 10).	**DIM A$(10,5)**
DOKE	Almacena los valores V en las posiciones X y X+1. y INT(V/256) va en X+1, y el restante en X.	**DOKEX,V**
DRAW	Dibuja un vector desde el actual cursor al cursor actual más X,Y.FB es 0-3.	**DRAW X,Y,FB**
END	Acaba el programa.	**END.**
EXP	Retorna el exponente natural de 0.	
EXPLODE	Produce un sonido predefinido.	**EXPLODE**
FALSE	Retorna un valor de 0.	
FILL	Llena las casillas del caracter A con las filas de B con el valor N. Hay 200 filas y 40 casillas de caracteres. N debe ser un entero entre el 0 y 127.	**FILL B,A,N**

FN	Produce el resultado de una función predefinida.	**PRINT FNA(X)**
FOR..TO	Crea un bucle que repite todas las líneas del programa entre FOR y NEXT. STEP determina la medida incremental. Si se omite, usa 1.	**FOR N=1 TO NEXT N**
FRE	Retorna la cantidad de memoria disponible en bytes.	**FRE(0)**
	También:- Fuerza la limpieza del área de variables en memoria.	**FRE("")**
GET	Sincroniza el teclado y espera que se pulse una tecla.	**GET A$**
GOSUB	Hace que la rama de un programa vaya al número especificado de línea.	**GOSUB 1000**
GOTO	Rama incondicional que va al número especificado de línea.	**GOTO 4000**
GRAB	Asigna el área en la memoria desde #9800 a #400(48K) o desde #1800 a #3400(16K) para usar RAM (Vea el mapa de la memoria).	**GRAB**
HEX$	Imprime el valor de V como un número hexadecimal.	**PRINT HEX$(V)**
HIMEM	Menos capacidad de memoria para los programas en BASIC. La memoria de de arriba puede ser usada para el código de programas de máquina.	**HIMEM #8700**
HIRES	Conecta en modo de alta resolución. El fondo de la pantalla a negro, en segundo término a blanco, el cursor a 0,0. Las líneas de texto permanecen en color.	**HIRES**
IF/THEN	Si la expresión que sigue a IF es verdadera, entonces ejecuta todas las instrucciones que siguen a THEN. Si la expresión es falsa, entonces estas instrucciones son ignoradas y el programa ejecuta las instrucciones que siguen a ELSE. ELSE puede ser omitido.	**IF A> 10 THEN PRINT "OK"**

INK	Cambia el color de toda la pantalla. N es un entero 0-7.	**INK N**
INPUT	Para la ejecución del programa y espera una entrada antes de continuar.	**INPUT N$ ";A**
INT	Retorna el entero más grande menor que o igual al valor entre comillas.	**INPUT(Y+ 0,5)**
KEY$	Sincroniza el teclado. Continua la ejecución, tanto si se ha pulsado o no una tecla. X$ contiene el valor de cualquier tecla pulsada.	**X$=KEY$**
LEFT$	Retorna la parte izquierda de una string, N caracteres de longitud.	**L$=LEFT$ (A$,N)**
LEN	Retorna la longitud de una string.	**A=LEN(N$)**
LET	Asigna el valor a una variable. (Opcional).	**LET A=4**
LIST	Lista las líneas especificadas o todo el programa. La barra de espacio para el listado.	**LIST 100** **LIST** **LIST 50-80**
LLIST	Lista las líneas especificadas o todo el programa por la impresora.	**LLIST 100** **LLIST**
LN	Retorna logaritmo natural.	**A=LN(X-2)**
LOG	Retorna logaritmos en base decimal.	**B=LOG(Y+1)**
LORESN	Conecta en modo de alta resolución. La pantalla de texto se fija en negro. Cuando N=0, se usa el conjunto de caracteres estandard. Cuando N=1, se usa el grupo de caracteres alternos.	**LORES 0**
MID$	Retorna una subrutina empezando en el caracter A, de longitud L.	**A$=MID$**
MUSIC	Vea el Capítulo del Sonido.	
NEW	Borra el programa actual y todas las variables.	**NEW**
ON...	Se ramifica a una subrutina en el número especificado de la línea N.	**ON N GOSUB 2000, 3000**
ON... GOTO	Se ramifica al número especificado de la línea N.	**ON N GOTO 1000, 2000**
PAPER	Cambia el color de toda la pantalla. N es un entero 0-7.	**PAPER N**

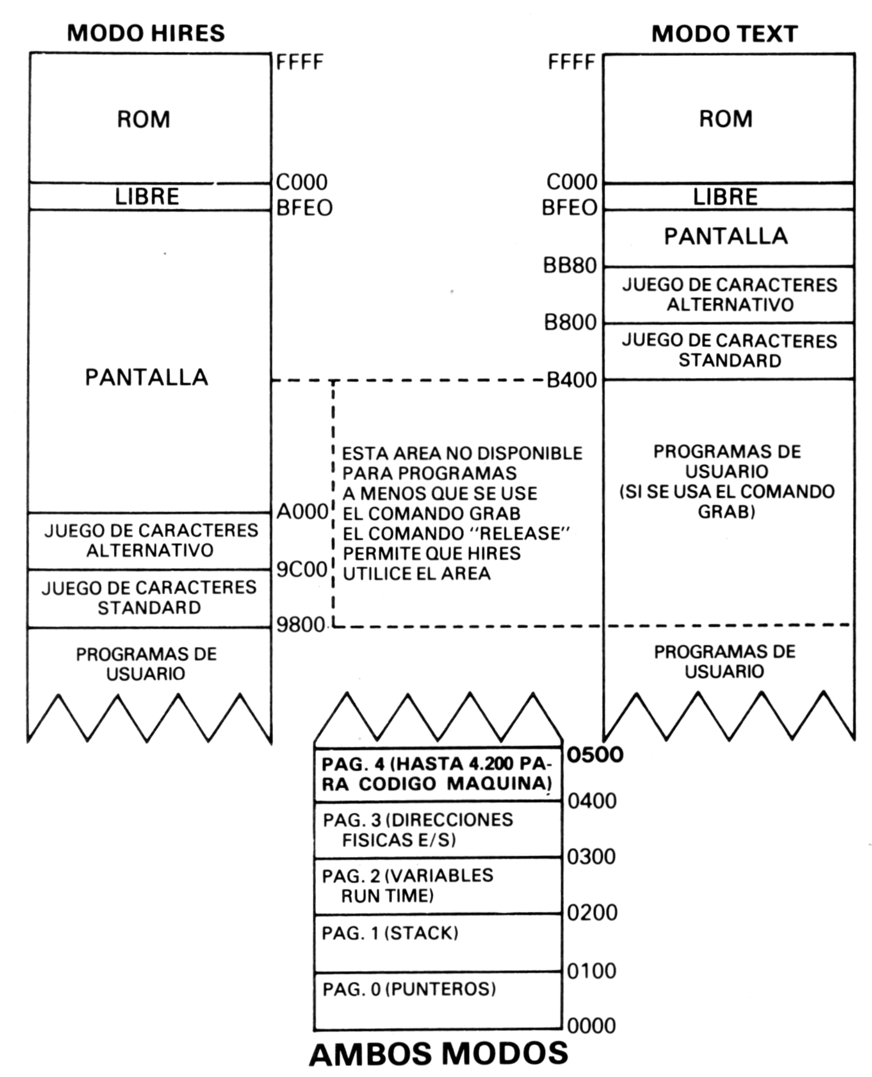
	Fija el registrador modelo para los comandos DRAW. X es un entero 0-255.	**PATTERN X**
PEEK	Retorna los contenidos de la posición X en memoria.	**A=PEEK(X)**
PI	Retorna el valor 3.14159265.	**PRINT2*PI**
PING	Produce un sonido predefinido.	**PING**
PLAY	Vea el Capítulo del Sonido.	
PLOT	Dibuja un caracter en la pantalla LORES o TEXT usando las coordinadas X+Y.	**PLOTX,Y,"X"** **PLOTX,Y,A$**
POINT	Retorna 0 si el pixel especificado es el fondo de la pantalla y -si el pixel está en segundo término. X es el valor absoluto de X (0-239) Y es el valor absoluto de Y (0-199)	**POINT(X,Y)**
POKE	Almacena el valor de V en la posición N en la memoria. V es un entero 0-255.	**POKE N,V**
POP	Hace que la dirección de un RETORNO deje un stack de direcciones de RETORNOS. El siguiente RETORNO que se encuentra a continuación de POP se ramifica a una sentencia detrás del segundo GOSUB ejecutado más recientemente.	**POP**
POS	Retorna la actual posición horizontal del cursor.	**A=POS**
PRINT	Imprime números, variables y strings en la pantalla. ? puede usarse en lugar del PRINT.	**"HELLO"** **PRINT N;A$**
PULL	Llama una dirección desde el stack en los bucles REPEAT. Vea POP.	**PULL**
READ	Lee el siguiente elemento en la lista DATA, y lo asigna a una variable especificada.	**READ A$,N**
	Asigna el área descrita en el comando GRAB a la pantalla de HIRES.	**RELEASE**
REM	Permite poner comentarios en las líneas del programa. Todo lo que va a continuación de REM se ignora.	**REM IGNORE THIS**

REPEAT	Crea un bucle para repetir todas las líneas del programa hasta la sentencia UNTIL. Verifica la sentencia en el UNTIL. Si es falso, repite el bucle. Sino, continua la ejecución en la siguiente línea del programa.	
	Fija el puntero READ al primer elemento en las líneas DATA.	**RESTORE**
	Hace retornar al computador a la sentencia que sigue al GOSUB más reciente.	**RETURN**
RIGHT$	Retorna la parte derecha de una string, N caracteres de longitud.	**R$=RIGHT$ (A$,N)**
RND	Retorna un número pseudi-random. Si X>=1, entonces el número está entre el 0 y el 1. Si X=0, entonces se produce el número que ha sido generado más recientemente. Si X <0 entonces el número que se produce es el mismo para cada X.	**A=RND(1)*6**
RUN	Ejecuta un programa en BASIC desde la línea N, o desde la línea más baja si N no está especificada. También borra todas las variables.	**RUN 200**
SCRN (X,Y)	Retorna el código ASCII para el caracter en la posición X,Y en modos LORES y TEXT.	
SGN	Retorna -1 si el argumento es negativo, 0 si es cero y 1 si es positivo.	**Z=SGN(X-Y)**
SHOOT	Produce un sonido predefinido.	**SHOOT**
SIN	Retorna el seno del ángulo N. N debe estar en radianes.	**A=SIN(N)**
SOUND	Vea el Capítulo del Sonido.	
SPC	Imprime N espacios en la pantalla. N es un entero 0-255.	**PRINT "HO" SPC(N) "HUM"**
SQR	Retorna la raiz cuadrada de N.	**A=SQR(N)**
STOP	Para la ejecución de un programa.	**STOP**
STR$	Convierte una expresión numérica dentro de una string.	**N$=STR$(N)**

TAB	Mueve la posición **PRINT** N lugares desde la izquierda de la pantalla.	**PRINT TAB(N)**
TAN	Retorna la tangente del ángulo N. N debe estar en radianes.	**"HELLO"** **A=TAN(N)**
TEXT	Conecta en modo texto.	**TEXT**
TROFF	Desconecta la pista de la función.	**TROFF**
TRON	Conecta una pista de la función.	**TRON**
TRUE	Retorna un valor de -1.	
USR	Pasa el valor entre comillas a una subrutina de punto flotante. Ver el Capºitulo 13.	**USR(N)**
VAL	Retorna el valor numérico de una string N$.	**A=VAL(N$)**
WAIT	Pausa condicional. N=10 msegs.	**WAIT (N)**
ZAP	Produce un sonido predefinido.	**ZAP**

Apéndice A

MAPA DE MEMORIA DEL ORIC 1 (40K)



AMBOS MODOS

N.B.

1) Para el 16K todas las direcciones (excepto ROM) son menos 8000 (hexa)
2) Todas las direcciones en hexa.

Caracteres de Control - todos los disponibles desde el teclado o a través de las sentencias **PRINT.**

1.
CTRL T - Mayúsculas/minúsculas
CTRL P - Impresora
CTRL F - Klick del teclado ON/OFF
CTRL D - Doble altura de letra ON/OFF
CTRL Q - Cursor
CRTL S - Parar listado en pantalla
CTRL - Columna protegida (la que está más a la izquierda)

2. Formato de caracteres de pantalla
CTRL J - Avance de línea
CTRL L - Borra la pantalla
CTRL M - Retorno de carro
CTRL N - Borra las filas

Para usar sentencias **Print** use

PRINT CHR$(x)

donde **x** es un número, **A=1, B=2**, etc., o sea:

CTRL D= CHR$(4)

Apéndice C

Atributos

b₆	Ø		Ø	
b_5	Ø		Ø	
b_4	Ø		1	
b_3 b_2 b_1 b_0				
Ø	LETRAS NEGRO	@	FONDO NEGRO	P
1	ROJO	A	ROJO	Q
2	VERDE	B	VERDE	R
3	AMARILLO	C	AMARILLO	S
4	AZUL	D	AZUL	T
5	MAGENTA	E	MAGENTA	U
6	CYAN	F	CYAN	V
7	BLANCO	G	BLANCO	W
8	SH/ST STD	H	TEXT 60HZ	X
9	SH/ST ALT	I	TEXT 60HZ	Y
A	DH/ST STD	J	TEXT 50HZ	Z
B	DH/ST ALT	K	TEXT 50HZ	{
C	SH/FL STD	L	GRA 60HZ	|
D	SH/FL ALT	M	GRA 60HZ	}
E	DH/FL STD	N	GRA 50HZ	~
F	DH/FL ALT	O	GRA 50HZ	←

escape character

50 Hz applicable in U.K.
60 Hz applicable in U.S.
Misuse may cause temporary loss of screen synchronization.

SH = ALTURA SENCILLA
DH = DOBLE ALTURA
ST = FIJO
FL = INTERMITENTE
GRA = GRAFICOS DE PUNTOS
STD = JUEGO DE CARACTERES STANDARD
ALT = JUEGO DE CARACTERES DEL USUARIO

CODIGOS A.S.C.I.I. (decimal)

Código	Caracter	Código	Caracter
32	Space	79	O
33	!	80	P
34	“	81	Q
35	#	82	R
36	$	83	S
37	%	84	T
38	&	85	U
39	’	86	V
40	(	87	W
41	)	88	X
42	*	89	Y
43	+	90	Z
44	,	91	[
45	-	92	\
46	.	93	]
47	/	94	↑
48	0	95	£
49	1	96	©
50	2	97	a
51	3	98	b
52	4	99	c
53	5	100	d
54	6	101	e
55	7	102	f
56	8	103	g
57	9	104	h
58	:	105	i
59	;	106	j
60	<	107	k
61	=	108	l
62	>	109	m
63	?	110	n
64	@	111	o
65	A	112	p
66	B	113	q
67	C	114	r
68	D	115	s
69	E	116	t
70	F	117	u
71	G	118	v
72	H	119	w
73	I	120	x
74	J	121	y
75	K	122	z
76	L	123	{
77	M	124	|
78	N	125	}

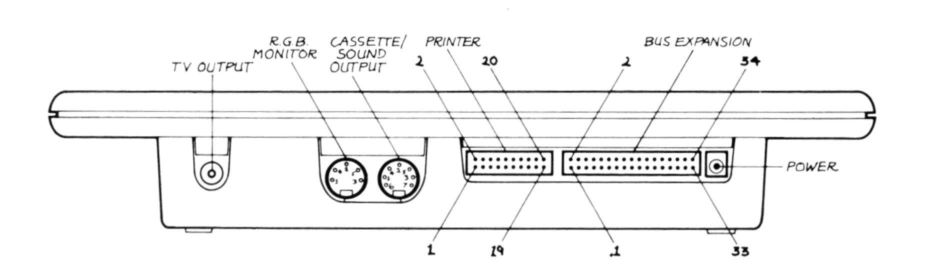
Tabla de conversión Decimal/Binaria/Hex

DEC.	BINARIA	HEX.
0	00000000	00
1	00000001	01
2	00000010	02
3	00000011	03
4	00000100	04
5	00000101	05
6	00000110	06
7	00000111	07
8	00001000	08
9	00001001	09
10	00001010	0A
11	00001011	0B
12	00001100	0C
13	00001101	0D
14	00001110	0E
15	00001111	0F
16	00010000	10
17	00010001	11
18	00010010	12
19	00010011	13
20	00010100	14
21	00010101	15
22	00010110	16
23	00010111	17
24	00011000	18
25	00011001	19
26	00011010	1A
27	00011011	1B
28	00011100	1C
29	00011101	1D

DEC.	BINARIA	HEX.
30	00011110	1E
31	00011111	1F
32	00100000	20
33	00100001	21
34	00100010	22
35	00100011	23
36	00100100	24
37	00100101	25
38	00100110	26
39	00100111	27
40	00101000	28
41	00101001	29
42	00101010	2A
43	00101011	2B
44	00101100	2C
45	00101101	2D
46	00101110	2E
47	00101111	2F
48	00110000	30
49	00110001	31
50	00110010	32
51	00110011	33
52	00110100	34
53	00110101	35
54	00110110	36
55	00110111	37
56	00111000	38
57	00111001	39
58	00111010	3A
59	00111011	3B

DEC.	BINARIA	HEX.	DEC.	BINARIA	HEX.
60	00111100	3C	94	01011110	5E
61	00111101	3D	95	01011111	5F
62	00111110	3E	96	01100000	60
63	00111111	3F	97	01100001	61
64	01000000	40	98	01100010	62
65	01000001	41	99	01100011	63
66	01000010	42	100	01100100	64
67	01000011	43	101	01100101	65
68	01000100	44	102	01100110	66
69	01000101	45	103	01100111	67
70	01000110	46	104	01101000	68
71	01000111	47	105	01101001	69
72	01001000	48	106	01101010	6A
73	01001001	49	107	01101011	6B
74	01001010	4A	108	01101100	6c
75	01001011	4B	109	01101101	6D
76	01001100	4C	110	01101110	6E
77	01001101	4D	111	01101111	6F
78	01001110	4E	112	01110000	70
79	01001111	4F	113	01110001	71
80	01010000	50	114	01110010	72
81	01010001	51	115	01110011	73
82	01010010	52	116	01110100	74
83	01010011	53	117	01110101	75
84	01010100	54	118	01110110	76
85	01010101	55	119	01110111	77
86	01010110	56	120	01111000	78
87	01010111	57	121	01111001	79
88	01011000	58	122	01111010	7A
89	01011001	59	123	01111011	7B
90	01011010	5A	124	01111100	7C
91	01011011	5B	125	01111101	7D
92	01011100	5C	126	01111110	7E
93	01011101	5D	127	01111111	7F

Pin Output Chart



BUS EXPANSION

MAP	1	2	ROMDIS
ØZ	3	4	RESET
I/O	5	6	I/O Control
R/W	7	8	IRQ
D2	9	1Ø	DØ
A3	11	12	D1
AØ	13	14	D6
A1	15	16	D3
A2	17	18	D4
D5	19	2Ø	A4
A5	21	22	D7
A6	23	24	A15
A7	25	26	A14
A8	27	28	A13
A9	29	3Ø	A12
A1Ø	31	32	A11
+5V	33	34	GND

PRINTER

STB	1	2	GND
DO	3	4	GND
D1	5	6	GND
D2	7	8	GND
D3	9	1Ø	GND
D4	11	12	GND
D5	13	14	GND
D6	15	16	GND
D7	17	18	GND
ACK	19	2Ø	GND

R.G.B.

1 — RED
2 — GREEN
3 — BLUE
4 — SYNC
5 — GND

CASSETTE/SOUND

1 Tape Out
2 GND
3 Tape In
4 Sound $_{8912}$
5 Sound
6 Relay Contact
7 Relay Contact

Funciones Matemáticas Especiales

Estas funciones no están directamente disponibles en el ORIC, pero pueden ser definidas usando **DEF FN.** Es decir:

DEF FN SC(X)=1/COS(X)

define el secante.

Secante:
SEC(X)=1/COS(X)

Cosecante:
CSC(X)=1/SIN(X)

Cotangente:
COT(X)=1/TAN(X)

Seno inverso:
ARCSIN(X)=ATN(X/SQR(-X*X+1)

Coseno inverso:
ARCSEC(X)=ATN(X/SQR(-X*X+1))+1.5708

Secante inverso:
ARCSEC(X)=ATN(SQR(X*X-1))+(SGN(X)-1)*1.5708

Cosecante inverso:
ARCCSC(X)=ATN(1/SQR(X*X-1))+(SGN(X)-1)*1.5708

Cotangente inversa:
ARCCOT(X)=-ATN(X)+1.5708

Seno hiperbólico:
SINH(X)=EXP(X)-EXP(-X))/2

Coseno hiperbólico:
COSH(X)=(EXP(X)+EXP(-X))/2

Tangente hiperbólica:
TANH(X)=-EXP(-X)/(EXP(X)+EXP(-X))*2+1

Secante hiperbólico:
SECH(X)=2/(EXP(X)+EXP(-X))

Cosecante hiperbólica:
CSCH(X)=2(EXP(X)-EXP(-X))

Cotangente hiperbólica:
COTH(X)=EXP(-X)/(EXP(X)-EXP(-X))*2+1

Seno hiperbólico inverso:
ARGSINH(X)=LOG(X+SQR(X*X+1))

Coseno hiperbólico inverso:
ARGCOSH(X)=LOG(+SQR(X*X-1))

Tangente hiperbólica inversa:
ARGTANH(X)=LOG((1+X)/1-X))/2

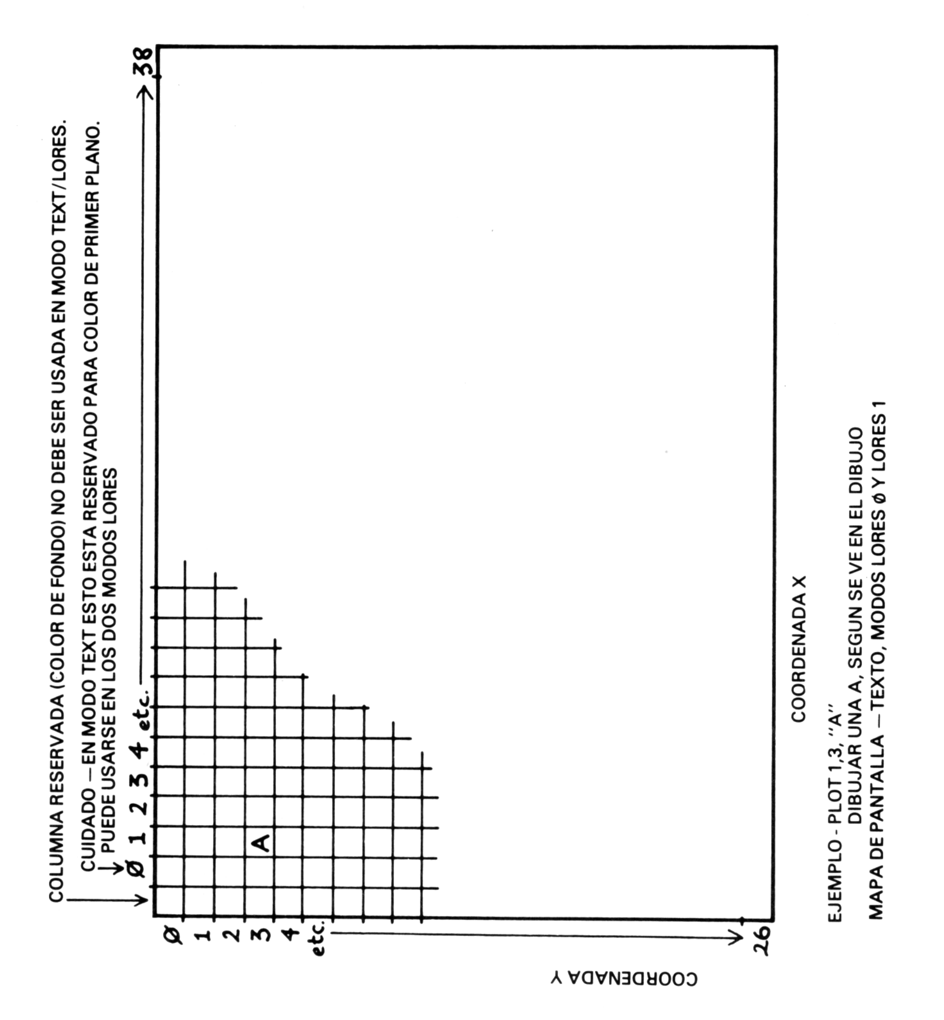
Secante hiperbólico inverso:
ARGSECH(X)=LOG((SQR(-X*X+1)+1)/X

Cosecante hiperbólico inverso:
ARGCOTH(X)=LOG((X+1)/(x-1))/2

A Módulo B:
MOD(A)=INT((A/B-INT(A/B))*B+0.05)*SGN(A/B)

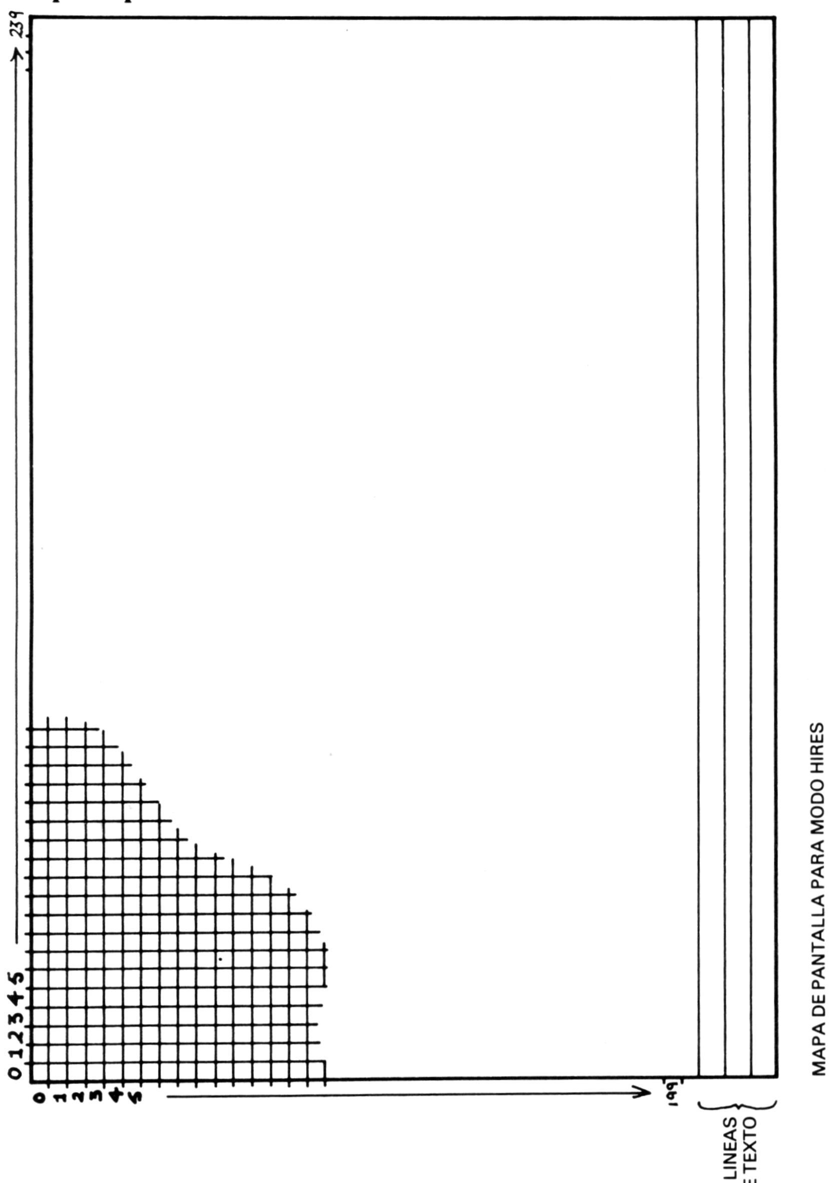
Apéndice H

Mapa de pantalla de texto



Apéndice I

Mapa de pantalla alta resolución



Códigos de Error

Si el ORIC no puede mantener un comando o información, entonces aparecerá un mensaje de error. Será seguido por el número de línea donde sucedió el error si estaba en un programa. Estos son los códigos posibles y sus significados.

1.) **CAN'T CONTINUE**
Intenta continuar un programa después de que se ha añadido una línea o borrado.

2.) **DISP TYPE MISMATCH**
Intenta DIBUJAR en modo de TEXTO o un problema similar.

3.) **DIVISON BY ZERO**
Difícil, ¡incluso con el ORIC!

4.) **FORMULA TOO COMPLEX**
Más de dos sentencias **IF/THEN** en la misma línea.

5.) **ILLEGAL DIRECT**
Una sentencia como **DATA** o **INPUT** que ha sido usada como un comando directo desde el teclado.

6.) **ILLEGAL QUANTITY**
Parámetro fuera de rango, es decir **SQR(-1)**

7.) **NEXT WITHOUT FOR**
Auto-explicativo ((¡esto es lo que uno desea!)

8.) **OUT OF DATA**
Intenta LEER después del final del listado **DATA.**

9.) **OUT MEMORY**
Auto-explicativo, pero también puede ser causado por más de 16 bucles **FOR...NEXT/TO** anidados o subrutinas.

10.) **OVERFLOW**
Un número mayor que $1.70141*10^{38}$ se ha producido durante una calculación.

11.) **REDIM'D ARRAY**
Intenta redimensionar una matriz previamente dimensionada.

12.) **RETURN WITHOUT GOSUB**
Auto-explicativo.

213.) **STRING TOO LONG**
Las strings deben de ser menos de 255 caracteres de longitud.

14.) **BAD SUBSCRIPT**
Cuando se ha intentado hacer referencia a un elemento que no existe, es decir, **LET A(24,25)**=Z cuando **A** ha sido dimensionada usando **DIM A(4,4).**

15.) **SYNTAX ERROR**
Puntuación incorrecta o falta de paréntesis, etc.

16.) **TYPE MISMATCH**
Cuando se ha intentado asignar una string a una variable numérica o vice versa.

17.) **UNDEF'D STATEMENT**
Cuando se ha intentado acceder a un número de línea que no existe usando GOTO, THEN o GOSUB.

18.) **UNDEF'D FUNCTION**
Cuando se ha intentado usar una función que no ha sido previamente definida.

19.) **REDO FROM START**
Cuando se ha intentado entrar una string cuando se pedía un número. Retrocede al comando **INPUT.**

20.) **BAD UNTIL**
Cuando se ha alcanzado un **UNTIL** sin haber encontrado anteriormente una sentencia **REPEAT.**

I.G. NTRA. SRA. DE LOS REYES 9 BARCELONA 23 D. L. B - 15227 / 83


D.S.E. S.A.
DISTRIBUIDORA DE SISTEMAS ELECTRONICOS, S. A.
Comte d'Urgell, 118 - BARCELONA-11

